

Le BASIC et ses Fichiers

Tome 2
programmes

Jacques Boisgontier

Editions du P.S.I.

Le BASIC
et ses FICHIERS
Tome 2
programmes

Dans la même collection

Programmer en Assembleur — Alain Pinaud
Programmer en BASIC — Michel Plouin
Le BASIC et ses fichiers — Jacques Boisgontier
Programmer en L.S.E. — Stéphane Berche et Yves Noyelle
Programmer en PASCAL — Daniel-Jean David et Jean-Luc Deschamps
Comment programmer — Jean-Claude Barbance
La découverte de l'Apple soft — Frédéric Lévy et Dominique Schraen
La pratique de l'Apple II - volume I — Nicole Bréaud-Pouliquen
La pratique de l'Apple II - volume II — Nicole Bréaud-Pouliquen
La pratique du LX500 - volume I — Alain Séméteys et Francis Vasse
La pratique du Sharp MZ-80K - volume I — Jean-Pierre Lhoir
La découverte du P.E.T. — Daniel-Jean David
La pratique du P.E.T./C.B.M. - volume I — Daniel-Jean David
La pratique du P.E.T./C.B.M. - volume II — Daniel-Jean David
La pratique du TRS-80 - volume I — Pierre Giraud et Alain Pinaud
La pratique du TRS-80 - volume II — Pierre Giraud et Alain Pinaud
La pratique du TRS-80 - volume III — Pierre Giraud et Alain Pinaud
Comprendre les microprocesseurs — Roland Dubois

Collection guides pratiques

La réalisation des programmes — Michel Benelfoul

ISBN : 2-86595-023-9

Imprimé en France

Le BASIC et ses FICHIERS
Tome 2
programmes

par
Jacques Boisgontier



Editions du P.S.I.
1981

P R E S E N T A T I O N

Pour la plupart, les programmes présentés sont à vocation professionnelle. Comme pour le volume 1, ils concernent les matériels disposant du BASIC MICROSOFT : TRS-80 et systèmes fonctionnant sous CPM.

Cet ouvrage fera gagner un temps appréciable à tous ceux qui doivent assurer une gestion en "temps réel" de leurs fichiers ; les méthodes d'accès par clé (Hash-Code et Accès Indexé) déjà abordées dans le volume 1 y sont traitées de façon plus approfondie.

Vous sont également proposés un générateur de saisie d'écran qui mettra en valeur vos programmes, un tri rapide, ainsi que de nombreux programmes 'types' en gestion : édition automatique, interrogation de fichier, etc...

Structurés et commentés, ces programmes seront aisément adaptables à vos problèmes.

SOMMAIRE

(*) Disponibles sur disquette TRS-80, chez Graphic - 14, avenue Pasteur - 93100 Montreuil - France.

CHAPITRE 1

LES FICHIERS A ACCES DIRECT

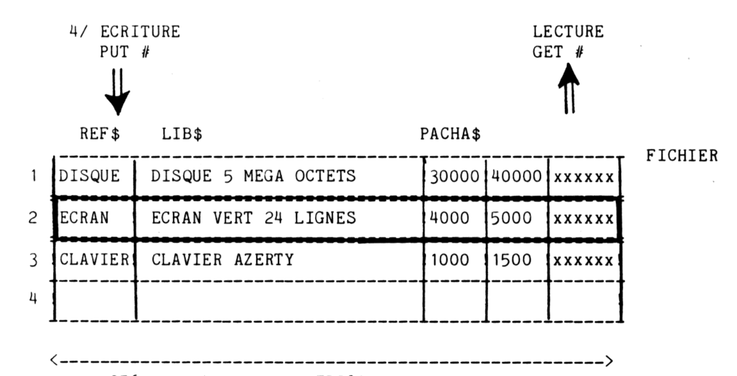
RAPPELS SUR LES FICHIERS A ACCES DIRECT

Les fichiers à accès direct considérés sont ceux du Basic Microsoft. Un fichier à ***accès direct*** (**RANDOM**) est une collection d'enregistrements de même longueur repérés par un numéro (1,2,3,...). La longueur de ceux-ci est soit fixée par le système (256 pour le TRS-80), soit choisie par l'utilisateur au moment de l'ouverture du fichier (MICROSOFT 5.).

- ☐ OUVERTURE DU FICHIER par OPEN
- ☐ DESCRIPTION DES ZONES DE LA MEMOIRE TAMPON PAR FIELD#,..
- ☐ REMPLISSAGE DES ZONES par LSET-RSET

1 enreg a la fois

ECRAN	ECRAN VERT 24 LIGNES	4000	5000	xxxxxxxx	MEMOIRE TAMPON (BUFFER)



OUVERTURE D'UN FICHIER : **OPEN #numéro fichier,"R","nom du fichier"**

Précisons d'abord que pour accéder à un fichier, il faut l'ouvrir par une instruction **'OPEN'**

```
60 OPEN 'R',1,"STOCK"
```

R spécifie le type de fichier (RANDOM)

1 représente un numéro choisi par le programmeur. Ce numéro servira dans la suite du programme à référencer le fichier 'STOCK'.

L'ouverture réserve une mémoire tampon (en mémoire centrale) où transiteront les informations du fichier.

Un fichier 'RANDOM' (aléatoire) est composé d'enregistrements référencés par un numéro (1 à 32000 par exemple).

Chacun de ces enregistrements correspond par exemple à une fiche client ou produit d'un fichier manuel.

TRS-80 :
- OPEN "R",#1,"STCK:1"   ' unité de disque 1
- Numéros autorisés:1-355
- Longueur fixe: 256 octets

MICROSOFT 5. :
- OPEN "R",#1,"B:STCK",50   '50:longueur des enregistrements
- Numéros autorisés:1-32000
- Longueur des enregistrements variable

FORMAT DES ENREGISTREMENTS : **FIELD #numéro fichier, longueur1 AS zone1,longueur2 AS zone2,...**

Un enregistrement est découpé en 'zones' (ou champs). Ainsi pour un enregistrement concernant un produit, nous avons différentes zones telles que la référence, le prix, le stock, etc.

La définition de la longueur des zones et de leur type est faite par une instruction **'FIELD'**

```
80 FIELD #1,12 AS REF$,25 AS LIB$,4 AS PACHA$,4 AS PVENTE$,..
```

12 caractères sont réservés pour la référence
25 caractères sont réservés pour le libellé
4 caractères sont réservés pour le prix d'achat

Nous verrons plus loin que plusieurs FIELD# peuvent être définis simultanément pour un même fichier.

LECTURE D'UN ENREGISTREMENT DEJA CREE : **GET #numéro fichier, numéro enregistrement**

La lecture en mémoire centrale d'un enregistrement déjà existant se fait par :

```
510 INPUT "Quel enregistrement? ";NE
520 GET #1,NE                ' NE  : Adresse de rangement
530 PRINT REF$,LIB$          ' Edition des zones REF$ et LIB$
```

Toutes les zones pour l'enregistrement lu peuvent alors être traitées (imprimées sur l'exemple).

CREATION D'ENREGISTREMENTS : **PUT #numéro fichier, numéro enregistrement**

Le transfert d'informations dans un fichier se fait par l'intermédiaire d'une mémoire tampon (buffer).

Les instructions **LSET** et **RSET** permettent de documenter les zones définies dans l'instruction FIELD#.

```
720 INPUT "Libelle? ";X$
730 LSET LIB$=X$              ' Affecte a LIB$ la valeur de X$
```

LSET transfère la valeur de X$ (entrée par INPUT) dans la zone LIB$ de la mémoire en cadrant à gauche, d'où le L (left) de LSET, RSET cadre à droite (Right).

Si 25 caractères ont été prévus pour la zone LIB$ et que le libellé a une longueur de 15, les 10 positions inoccupées à droite de LIB$ sont complétées par des espaces. Il est interdit de faire directement INPUT "Libellé ? ";LIB$.

Lorsque les valeurs des différentes zones définies dans FIELD ont été affectées, le transfert de la mémoire tampon dans le fichier sur disque se fait par :

```
570 PUT #1,NE  'Ecriture dans l'enregistrement de numéro NE
```

où #1 représente le numéro de fichier défini à l'ouverture et
NE l'adresse où est rangé l'enregistrement.

Le rangement d'un nouvel article se fait généralement en utilisant la fonction LOF (numéro fichier) qui fournit le nombre d'enregistrements d'un fichier. On fait donc le rangement en LOF(1)+1.

```
660 NE  =LOF(1)+1            ' initialisation de la memoire tampon
670 GET #1,NE                ' avec des valeurs ASCII nulles
690 INPUT "Reference? ";X$
710 LSET REF$=X$             ' Remplissage de la zone REF$
820 PUT #1,NE                ' Ecriture dans le fichier
```

L'instruction GET #1,NE (sachant que NE =LOF(1)+1) lit dans la mémoire tampon (buffer) l'enregistrement de numéro NE contenant des valeurs ASCII nulles et initialise ainsi les différentes zones définies dans FIELD# avec des valeurs ASCII nulles. Ceci afin d'éviter qu'au moment de l'écriture par PUT #, les zones, à qui il n'aurait pas été affecté de valeur à la saisie aient celles de l'enregistrement précédemment traité (en lecture ou écriture).

Un fichier n'est pas une table, on ne dispose, en mémoire centrale, que d'un enregistrement à la fois.

Remarque : l'initialisation de la mémoire tampon peut également être programmée ainsi : (MICROSOFT 5.)

```
10 FIELD #1,128 AS X$
20 FIELD #1,15 AS REF$,25 AS LIB$,....
   :
   :
100 LSET X$=STRING$(CHR$(0),128)
```

Erreur à ne pas commettre: il est interdit d'affecter directement une valeur à une variable définie dans FIELD # sans utiliser LSET ou RSET :

```
INPUT "Référence? ";REF$   'Interdit
```

La variable REF$ se trouverait désalouée de la mémoire tampon et serait considérée dans la suite du programme comme une variable normale.

DIFFERENTS TYPES DE ZONES

Les valeurs numériques font l'objet d'un traitement particulier : elles sont en effet compactées sur disque sous forme de chaînes de caractères.

Par exemple, des nombres entiers compris entre -32000 et +32000 ne nécessitent que 2 octets. Les types de zones ne sont pas définis explicitement dans l'instruction FIELD.

C'est par une instruction de conversion au moment du LSET que le compactage s'effectue.

```
LSET QVENDU$=MKI$(X)   'Compacte X sur 2 octets
```

La place réservée dans FIELD# pour QVENDU$ doit être de 2 octets.

A la lecture, la conversion inverse doit être effectuée par X=CVI(QVENDU$), (PRINT QVENDU$ n'aurait pas de signification).

Suivant les types de variables numériques, les conversions se font par :

ECRITURE	LECTURE		
MKI$	CVI	-32768<entiers<+32767	2 octets réservés dans FIELD
MKS$	CVS	simple précision	4 octets réservés dans FIELD
MKD$	CVD	double précision	8 octets
CHR$	ASC	0<entiers<255	1 octet

SUPPRESSION D'ENREGISTREMENTS

La suppression d'enregistrements n'existe pas explicitement. Le plus simple est de placer des valeurs ASCII nulles à l'intérieur de l'enregistrement à supprimer.

Ces valeurs nulles permettront plus tard de repérer les enregistrements inutilisés.

```
GET #1,LOF(1)+1        'Remplissage de la mémoire
                        tampon avec zéros ASCII
PUT #1, numéro à supprimer
```

La place disque de l'enregistrement supprimé n'est pas récupérée.

FIN DE FICHIER *:* **LOF(numéro fichier)**

TRS-80 : la fonction LOF permet, sur TRS-80, de connaître le numéro du dernier enregistrement d'un fichier (le numéro le plus élevé).

MICROSOFT 5.: La fonction LOF n'a plus le même sens et n'a guère d'utilité, elle représente en effet le nombre de blocs de 128 octets utilisés dans le dernier extent référencé (1 extent=128X128 octets).

Il faut donc gérer soi-même un compteur d'enregistrements ou mieux assurer l'allocation dynamique des enregistrements (cf. HASH-CODE et allocation dynamique).

FERMETURE DE FICHIER : CLOSE #numéro fichier

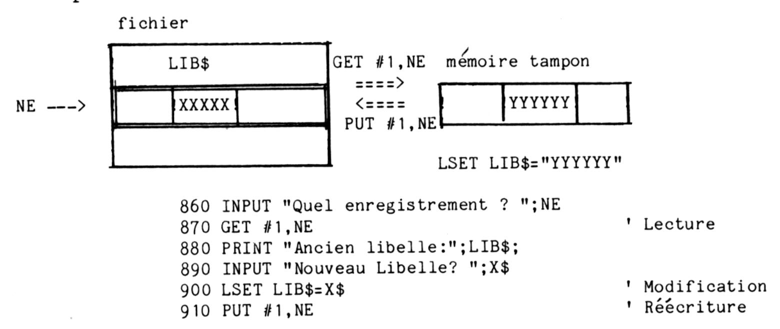
L'instruction CLOSE libère la mémoire tampon du fichier clos.

Le bloc de contrôle du fichier amené en mémoire centrale à l'OPEN et mis à jour en cas d'ajouts d'enregistrements n'est sauvegardé qu'à la fermeture qui est donc obligatoire (un DOS bien conçu devrait prévoir cette sauvegarde à chaque ajout).

MODIFICATION D'UN ENREGISTREMENT DEJA CREE

La modification d'une zone dans un enregistrement ne peut se faire que par l'intermédiaire de la mémoire tampon. Il faut d'abord lire l'enregistrement, puis modifier la zone et enfin transférer la mémoire tampon dans le fichier par l'instruction PUT #.

L'oubli de GET # avant LSET affecterait les zones qui ne sont pas modifiées.

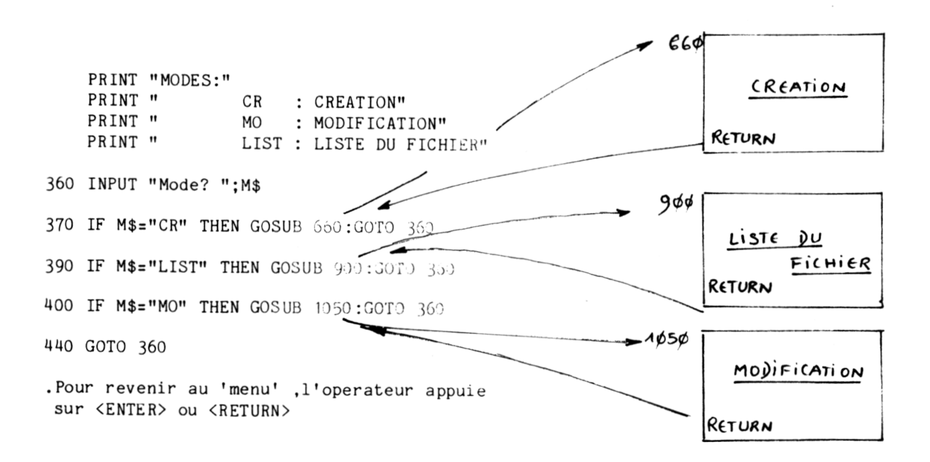



```
860 INPUT "Quel enregistrement ? ";NE
870 GET #1,NE                              ' Lecture
880 PRINT "Ancien libelle:";LIB$;
890 INPUT "Nouveau Libelle? ";X$
900 LSET LIB$=X$                           ' Modification
910 PUT #1,NE                              ' Réécriture
```

PROGRAMME DE SYNTHESE

NOTION DE MENU

Plutôt que d'écrire plusieurs programmes indépendants qu'il faudrait charger individuellement à chaque fois que l'un d'eux doit être exécuté, il est plus pratique qu'ils soient tous regroupés dans un seul programme et qu'un système d'aiguillage permette de sélectionner un sous-programme particulier.



Si l'opérateur répond à la question 'Mode? ' par 'CR', c'est le sous-programme de création qui est appelé.

DIFFERENTS MODES

Dans le programme présenté, sont réunies la création, la lecture et la modification d'enregistrements.

D'autres modes permettent d'éditer le fichier, l'inventaire des produits et la répartition des ventes.

LISTE DU FICHIER

La liste du fichier s'obtient par :

```
740 FOR I=1 TO LOF(1)          ' Pour BASIC 5. Cf EOF
750   GET #1,I
760   PRINT REF$,LIB$
770 NEXT I
```

REPARTITION DES VENTES

Il s'agit de classer par ordre d'importance décroissante les ventes réalisées sur différents produits. Pour cela, trois tables sont constituées par une lecture séquentielle du fichier :

. 1 table des ventes où est stocké pour chaque article le produit PRIX*QT VENDUE,
. 1 table des références,
. 1 table des libellés.

VNTE()	REF$()	LIB$()
150000	DISQUE	DISQUE 5 mega octets
40000	ECRAN	ECRAN VERT 24 LIGNES

Ces tables sont ensuite triées et enfin les résultats sont édités sous forme d'un tableau puis d'un histogramme.

```
10 ' INIT/BAS 23.2.81           FICHIER STOCK
40 '
50 CLEAR(1000)           ' Reservation pour l'espace chaines de caracteres
60 OPEN "R",#1,"STCK"
70 '
80 FIELD #1,12 AS REF$,25 AS LIB$,4 AS PACHA$,4 AS PVENTE$,4 AS QV$,4 AS STK$
90 '
100 '    DESCRIPTION DES ZONES DU FICHIER 'STOCK'
110 '
120 '  REF$     :REFERENCE                                  12 C
130 '  LIB$     :LIBELLE                                    25 C
140 '  PACHA$   :PRIX ACHAT (SIMPLE PRECISION)               4 C
150 '  PVENTE$  :PRIX VENTE (SIMPLE PRECISION)               4 C
160 '  QV$      :QUANTITE VENDUE (SIMPLE PRECISION)          4 C
170 '  STK$     :STOCK      (SIMPLE PRECISION)               4 C
180 '
190 '   ---------------------------------------------------
200 '  1 !ECRAN!ECRAN VERT  24 LIGNES    !4000  !5000 !     !
210 '   ---------------------------------------------------
220 '  2 !DISQUE!DISQUE 5 MEGA OCTETS    !30000 !40000!     !
230 '   ---------------------------------------------------
240 '     REF$    LIB$                  PACHA$  PVENTE$
250 '
260 DIM VNTE(50)           ' TABLE POUR VENTES
270 DIM REF$(50)           ' TABLE POUR REFERENCES
280 DIM LIB$(50)           ' TABLE POUR LIBELLES
290 '
300 PRINT TAB(10) "MODES:":PRINT
310 PRINT TAB(15) "CR   : CREATION "
320 PRINT TAB(15) "L    : LECTURE D'UN ARTICLE"
330 PRINT TAB(15) "MO   : MODIFICATION"
340 PRINT TAB(15) "FIN  : FIN (OBLIGATOIRE SI CREATION)"
350 '
360 PRINT:INPUT "MODE ";M$
370 IF M$="CR" THEN GOSUB 660:GOTO 300          ' Creation
380 IF M$="L" THEN GOSUB 510:GOTO 300           ' Lecture
390 IF M$="LIST" THEN GOSUB 900:GOTO 300        ' Liste du fichier
400 IF M$="MO" THEN GOSUB 1050:GOTO 300         ' Modification
410 IF M$="IS" THEN GOSUB 580:GOTO 300          ' Incrementation stock
420 IF M$="INV" THEN GOSUB 1920:GOTO 300        ' Inventaire
430 IF M$="REPV" THEN GOSUB 1340:GOTO 300       ' Repartition des ventes
440 IF M$="FIN" THEN CLOSE #1:STOP
450 GOTO 300
460 '
470 '
480 '
490 '=================================================================
500 '                                   LECTURE D'UN ARTICLE
510 PRINT:NE=0:INPUT "Quel enregistrement ";NE:IF NE=0 THEN RETURN
520 GET #1,NE
530 PRINT:PRINT REF$,LIB$
540 GOTO 510
550 '================================================================
570 '                                   INCREMENTATION STOCK
575 '
580 NE=0:INPUT "Quel enregistrement ";NE:IF NE=0 THEN RETURN
590 GET #1,NE:PRINT "STOCK=";CVS(STCK$);
600 INPUT "  Quel increment ";X
605 INPUT "OK O/N ";R$:IF R$<>"O" THEN 580
610 LSET STK$=MKS$(CVS(STK$)+X)
620 PUT #1,NE:GOTO 580
630 '
640 '==============================================================
```

```
646 '
650 '                                    CREATION D'UN ARTICLE
660 NE=LOF(1)+1                       ' Rangement en fin de fichier
670 GET #1,NE                         ' Initialisation du buffer
680 PRINT
690 X$="":INPUT "REFERENCE          ";X$
700 IF X$="" THEN RETURN              ' Test fin de mode creation
710 LSET REF$=X$                      ' Transfert de X$ DANS REF$
720 X$="":INPUT "LIBELLE            ";X$
730 LSET LIB$=X$
740 X=0:INPUT "PRIX D'ACHAT      ";X
750 LSET PACHA$=MKS$(X)
760 X=0:INPUT "PRIX VENTE        ";X
770 LSET PVENTE$=MKS$(X)
780 X=0:INPUT "QUANTITE VENDUE   ";X
790 LSET QV$=MKS$(X)
800 X=0:INPUT "STOCK             ";X
810 LSET STK$=MKS$(X)
820 PUT #1,NE
830 PRINT:PRINT "Article range en:";NE
840 GOTO 660
850 '
860 '
870 '==============================================================
880 '                              LISTE DU FICHIER 'STOCK'
890 '
900 LPRINT
910 LPRINT:LPRINT "LISTE DU FICHIER":LPRINT
920 LPRINT TAB(5) "REFERENCE" TAB(20) "LIBELLE" TAB(45)
     "PRIX D'ACHAT" TAB(58) "QT VENDUE"
930 LPRINT
940 FOR I=1 TO LOF(1)                  ' Tout le fichier
950   GET #1,I
960   IF ASC(REF$)=0 THEN GOTO 980     ' Enreg vide?
970   LPRINT I TAB(5) REF$; TAB(20) LIB$; TAB(45) CVS(PACHA$);
        TAB(58) CVS(QV$)
980 NEXT I
990 RETURN
1000 '==============================================================
1010 '                                  MODIFICATION D'UN ARTICLE
1020 '     ON AFFICHE LES ANCIENNES VALEURS.SI L'OPERATEUR NE
1030 '     VEUT PAS CHANGER CES VALEURS,IL APPUIE SUR 'ENTER'
1040 '
1050 PRINT:NE=0:INPUT "QUEL ENREGISTREMENT ";NE
1060 IF NE=0 THEN RETURN               ' Fin de mode si <ENTER>
1070 GET #1,NE
1080 PRINT
1090 PRINT "REFERENCE:";TAB(15);
1100 PRINT REF$; TAB(45); : X$="":INPUT X$:IF X$><"" THEN
       LSET REF$=X$
1110 PRINT "LIBELLE:" TAB(15)
1120 PRINT LIB$;:PRINT TAB(45) :X$="":INPUT X$:IF X$><"" THEN
       LSET LIB$=X$
1130 PRINT "PRIX D'ACHAT:";TAB(15);
1140 PRINT CVS(PACHA$);TAB(45):X=0:INPUT X:IF X><0 THEN
       LSET PACHA$=MKS$(X)
1150 PRINT "QUANTITE VENDUE:";TAB(15)
1160 PRINT CVS(QV$);TAB(30) :X=0:INPUT X:IF X><0 THEN
       LSET QV$=MKS$(X)
1170 PUT #1,NE
1180 GOTO 1050
1190 '==============================================================
```

```
1260 '
1270 '                REPARTITION DES VENTES
1280 '
1290 ' 1/ CONSTITUTION DE 3 TABLES:
1300 '          -TABLE DES VENTES                VNTE()
1310 '          -TABLE DES REFERENCES            REF$()
1320 '          -TABLE DES LIBELLES              LIB$()
1330 '
1340 NB=0:TTAL=0:TP=0       ' NB: NOMBRE DE PIECES
1350 '                      ' TTAL: TOTAL    - TP: TOTAL PARTIEL
1360 FOR I=1 TO LOF(1)
1370   GET #1,I:IF ASC(REF$)=0 GOTO 1440    ' Enreg vide?
1380   PRINT REF$
1390   NB=NB+1
1400   VNTE(NB)=CVS(PACHA$)*CVS(QV$)
1410   REF$(NB)=REF$
1420   LIB$(NB)=LIB$
1430   TTAL=TTAL+CVS(PACHA$)*CVS(QV$)
1440 NEXT I
1450 '------------------------------------------------------------
1460 '  2/ TRI DES TABLES    (TRI DU TYPE 'RIPPLE')
1470 '
1480 INV=0             ' TEMOIN D'INVERSION
1490 FOR I=1 TO NB-1
1500   IF VNTE(I+1)>VNTE(I) THEN
         X=VNTE(I):VNTE(I)=VNTE(I+1):VNTE(I+1)=X:
         X$=REF$(I):REF$(I)=REF$(I+1):REF$(I+1)=X$:
         X$=LIB$(I):LIB$(I)=LIB$(I+1):LIB$(I+1)=X$:INV=1
1510 NEXT I
1520 IF INV<>0 GOTO 1480
1530 '------------------------------------------------------------
1540 '   3/ EDITIONS RESULTATS
1550 '
1560 LPRINT:LPRINT "REPARTITION DES VENTES":LPRINT
1570 TP=0
1580 LPRINT "REFERENCE" TAB(15) "VENTES" TAB(35)
       "% TOTAL":LPRINT
1590 FOR I=1 TO NB
1600   TP=TP+VNTE(I)
1610   LPRINT REF$(I);TAB(15)
1620   LPRINT USING "#######.##";VNTE(I);
1630   LPRINT TAB(35) :LPRINT USING " ##.##";VNTE(I)/TTAL;
1640   LPRINT TAB(45):LPRINT USING " ##.##";TP/TTAL
1650 NEXT I
1660 '------------------------------------------------------------
1670 ' 4/                            HISTOGRAMME
1680 MX=0
1690 FOR I=1 TO NB                ' Recherche du plus grand
1700   IF VNTE(I)>MX THEN MX=VNTE(I)
1710 NEXT I
1720 ECH=10/MX                    ' Calcul de l'echelle
1730 '------------------- Edition
1740 LPRINT:LPRINT
1750 FOR I=1 TO NB
1760   LPRINT REF$(I);
1770   X=INT(VNTE(I)/MX*10)
1780   IF X<1 THEN LPRINT:GOTO 1800
1790   FOR J=1 TO X:LPRINT "*";:NEXT J:LPRINT  ' Edition d'une ligne
1800 NEXT I
1810 LPRINT TAB(20):LPRINT USING "ECHELLE: ##.#####";ECH
1820 RETURN
1830 '=====================================================
```

```
1910 '                                    INVENTAIRE
1920 TV#=0                     ' Total en double precision
1930 LPRINT
1940 LPRINT :LPRINT "INVENTAIRE":LPRINT
1950 LPRINT TAB(3) "REFERENCE" TAB(17) "LIBELLE" TAB(44)
       "PRIX ACHAT" TAB(55) "VALEUR STOCK"
1960 LPRINT
1970 '
1980 FOR I=1 TO LOF(1)                     ' Tout le fichier
1990   GET #1,I
2000   IF ASC(REF$)=0 THEN GOTO 2090       ' Enreg vide?
2010   Q=CVS(STK$):P=CVS(PACHA$)
2020   Q#=VAL(STR$(Q)):P#=VAL(STR$(P)) ' Conversion en double precision
                       cf BASIC ET SES FICHIERS P17
2030 '
2040   V#=Q#*P#
2050   TV#=TV#+V#
2060   LPRINT I TAB(3) REF$;TAB(17);LIB$;
2065   LPRINT TAB(45):LPRINT USING "#####.##";CVS(PACHA$);
2070   LPRINT TAB(56):LPRINT USING "#####.##";V#;
2080   LPRINT TAB(62):LPRINT USING "######";Q
2090 NEXT I
2100 LPRINT
2110 LPRINT "TOTAL=";TV#
2120 RETURN
2130 '=======================================================
```

LISTE DU FICHIER

	REFERENCE	LIBELLE	PRIX D'ACHAT	QT VENDUE
1	ECRAN	ECRAN VERT 24 LIGNES	4000	10
2	DISQUE	DISQUE 5 MEGA OCTETS	30000	5
3	CLAVIER	CLAVIER AZERTY	1000	10
4	IMPRIMANTE	IMPRIMANTE 300 LIGNES/MN	5000	4
5	DISQUETTE	DISQUETTES 8 POUCES	20	1000

REPARTITION DES VENTES

REFERENCE	VENTES	% TOTAL	
DISQUE	150000.00	0.63	0.63
ECRAN	40000.00	0.17	0.79
IMPRIMANTE	20000.00	0.08	0.88
DISQUETTE	20000.00	0.08	0.96
CLAVIER	10000.00	0.04	1.00

```
DISQUE      **********
ECRAN       **
IMPRIMANTE  *
DISQUETTE   *
CLAVIER
                    ECHELLE:  0.00007
```

INVENTAIRE

	REFERENCE	LIBELLE	PRIX ACHAT	VALEUR STOCK	
1	ECRAN	ECRAN VERT 24 LIGNES	4000.00	16000.00	4
2	DISQUE	DISQUE 5 MEGA OCTETS	30000.00	60000.00	2
3	CLAVIER	CLAVIER AZERTY	1000.00	5000.00	5
4	IMPRIMANTE	IMPRIMANTE 300 LIGNES/MN	5000.00	10000.00	2
5	DISQUETTE	DISQUETTES 8 POUCES	20.00	4000.00	200

TOTAL= 95000

SUPPRESSION D'UN FICHIER : **KILL "nom de fichier"**

La suppression d'un fichier par programme s'écrit :

```
10 KILL "STCK"   'Le fichier 'STCK' doit être clos.
```

La place disque occupée par le fichier est récupérée par le système pour une allocation ultérieure.

BLOC DE CONTROLE
DU FICHIER 'STCK'

1 GRANULE = 1024 OCTETS

* Lors d'une suppression de fichier, les différents granules qui lui étaient affectés sont récupérés par le système pour d'autres fichiers

BIT MAP D'OCCUPATION DISQUE (POUR L'ENSEMBLE DES FICHIERS)

INITIALISATION DE FICHIER

Sur TRS-80, l'écriture dans un fichier vide au numéro N crée, en plus de l'enregistrement N, les N-1 enregistrements précédents, avec des valeurs quelconques. Par conséquent, rien ne permet de distinguer les enregistrements déjà créés de ceux qui ne le sont pas encore.

Aussi, afin d'éviter cela, les enregistrements d'un fichier peuvent-ils être initialisés avec des valeurs ASCII nulles.

```
10 INPUT "Nom de fichier? ";NF$
20 INPUT "Combien d'enregistrements a initialiser? ";NE
30 '
40 OPEN "R",#1,NF$
50 FIELD #1,255 as X$
60 '
70 GET #1,LOF(1)+1                    ' Valeurs ASCII nulles dans buffer
80 '
90 FOR I=1 TO NE
100    PUT #1,I
110 NEXT I
120 CLOSE #1
```

CONSEILS POUR LA MISE AU POINT

Comment procéder lorsque les informations que l'on pense avoir stockées dans un fichier ne sont pas retrouvées ?

1/ Par insertion d'instructions dans le programme :

```
100 LSET LIB$="XXXXXXXXX"
110 PUT #1,X                    ' Ecriture dans le fichier
-------------------------
112 GET #1,X                    ' On lit immediatement.Pour test seulement
114 PRINT LIB$,REF$,X
-------------------------
120 ..suite du pgm ..........
```

Juste après avoir écrit dans le fichier par PUT #, l'enregistrement est lu par GET #et les zones définies dans FIELD# sont visualisées.

2/ <u>Lecture dans le fichier en 'MODE DIRECT'</u>.

Après l'écriture par PUT#, on interrompt le programme (par contrôle C ou break), puis on écrit en mode direct :

```
GET #1,X                    ' Mode direct
PRINT LIB$,REF$
```

DECLARATION DE TABLEAUX PAR FIELD#

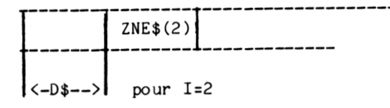
Des tableaux à une ou plusieurs dimensions se déclarent de deux façons :

a/ Les zones ont des **longueurs différentes** :

```
FIELD #1,4 AS ZNE$(1),5 AS ZNE$(2),3 AS ZNE$(3),......
```

b/ Si tous les éléments du tableau ont **la même longueur** :

```
100 FOR I=1 to 5
110    FIELD #1,4*(I-1) AS D$,4 AS ZNE$(I)
120 NEXT I
```



La variable D$ ne sert qu'à positionner les éléments de la table ZNE$(). Pour i=2 par exemple, elle positionne ZNE$(2) en 5.

- 2 tables à 1 dimension :

```
100 FOR I=1 to 5
110    FIELD #1,(4+2)*(I-1) AS D$,4 AS X$(I),2 AS Y$(I)
120 NEXT I
```

- 1 table à 2 dimensions :

```
100 FOR J=1 TO 5
110   FOR I=1 TO 3
120     FIELD #1,(3*2)*(J-1) AS D$,2*(I-1) AS D$,2 AS X$(J,I)
130   NEXT I
140 NEXT J
```

```
--------------------------------------------------------------
|X$(1,1) | X$(1,2) | X$(1,3) | X$(2,1) | X$(2,2) |      !
----------------------------------------------------------------
| 2        2         2       | 2
|                            |
|<------ 3*2 --------------->|
```

CHAMPS MULTIPLES

Pour un même fichier, plusieurs FIELD (champs) peuvent être définis.

```
100 FIELD #1,12 AS REF$    ,25 AS LIB$    ,4 AS PACHA$,...
110 FIELD #1,12 AS ZNE$(1),25 AS ZNE$(2),4 AS ZNE$(3),..
```

La référence par exemple est connue à la fois sous le nom de REF$ et de ZNE$(1) aussi bien en lecture qu'en écriture : PRINT REF$ est équivalent à PRINT ZNE$(1). Ceci permet d'accéder à une zone soit par un indice, soit par un nom plus mnémonique.

Cette facilité permet également de définir plusieurs structures pour un même fichier. Sur l'exemple, l'enregistrement 1 contient le nombre d'enregistrements du fichier et la date.

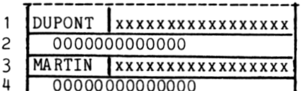
1	Nb enregistrements	Jour	Mois	An
2	R5	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx		
3	CX	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx		

REORGANISATION DE FICHIER RANDOM

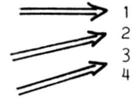
Si la récupération des enregistrements inutilisés dans un fichier n'est pas prévue par programme, il faut procéder périodiquement à un 'retassage' du fichier en supprimant les enregistrements vides.

Le plus simple, pour indiquer qu'un enregistrement n'est plus utilisé, est d'y placer des valeurs ASCII nulles.

FICHIER SOURCE



1	DUPONT	xxxxxxxxxxxxxxxxx
2	0000000000000	
3	MARTIN	xxxxxxxxxxxxxxxxx
4	0000000000000	
5	DURAND	xxxxxxxxxxxxxxxxx



FICHIER REORGANISE

1	DUPONT	xxxxxxxxxxxxxxx
2	MARTIN	xxxxxxxxxxxxxxx
3	DURAND	xxxxxxxxxxxxxxx
4	ABRAHAM	xxxxxxxxxxxxxxx

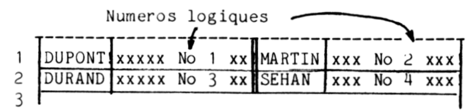
```
 5 INPUT "Nom fichier Source? ";NF$
10  OPEN "R",#1,NF$                          ' Fichier source
20 OPEN "R",#2,NF$+"X":CLOSE #2
25 KILL NF$+"X":OPEN "R",#2,NF$+"X"          ' Fichier reorganise
30 '
40  FIELD #1,128 AS X1$
50  FIELD #2,128 AS X2$
60 '
65 PRINT "Nom du fichier reorganise:";NF$+"X"
70 X2=0                        ' X2:Pointeur fichier reorganise
100 FOR X1=1 TO LOF(1)         ' Lecture de tout le fichier source
110    GET #1,X1
120    IF ASC(X1$)=0 THEN 200
130    LSET X2$=X1$
140    X2=X2+1:PUT #2,X2       ' Rangement dans le fichier reorganise
200 NEXT X1
210 '
220 CLOSE #1,#2
```

S'il existe des pointeurs vers le fichier réorganisé (dans d'autres fichiers), ils doivent bien sûr être mis à jour.

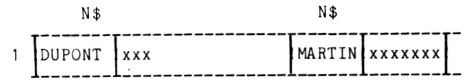
ENREGISTREMENTS LOGIQUES

Sur TRS-80, le choix d'une longueur de 256 caractères pour les enregistrements n'est pas très judicieux, une longueur de 128 aurait mieux convenu.

Comment définir 2 enregistrements logiques par enregistrement physique ?



FIELD DYNAMIQUE

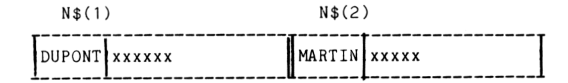


L'instruction FIELD permet de définir dynamiquement la position d'une variable dans la mémoire tampon :

```
10 INPUT "Quel enregistrement logique? ";NL
20 '
30 NP=INT((NL-1)/2)+1          ' NP:No physique
40 PS=((NL-1) MOD 2)+1         ' PS:Position dans l'enreg (1,2)
50 FIELD #1,128*(PS-1) AS D$,12 AS N$
60 GET #1,NP
70 INPUT "Nom? ";X$
80 LSET N$=X$
90 PUT #1,NP
100 GOTO 10
```

DECLARATION DE TABLEAUX

NØ devient une table NØ() à 2 éléments, (ainsi que les autres zones).




```
5 FIELD #1,15 AS N$(1),...,15 AS N$(2)
10 INPUT "Quel enregistrement logique? ";NL
20 '
30 NP=INT((NL-1)/2)+1
40 PS=((NL-1) MOD 2)+1
50 GET #1,NP
70 INPUT "Nom? ";X$
80 LSET N$(PS)=X$
90 PUT #1,NP
100 GOTO 10
```

SAISIE POUR FICHIER RANDOM

Une simple suite d'INPUT et de LSET permet de documenter les différentes zones d'un enregistrement. C'est ce que nous avions fait dans l'exercice précédent.

Cette méthode présente deux inconvénients :

- Dès que le nombre de zones à saisir devient important, l'écriture de la suite d'INPUT et de LSET est fastidieuse.
- En outre, en cours de saisie, on n'a pas la faculté de se positionner en arrière sur une zone qui aurait été mal documentée.

En revanche, le programme proposé le permet : nous définissons les différentes zones à saisir dans la mémoire tampon comme des éléments d'une table.

3 tables contiennent :

. les noms des zones NZ$()
. les longueurs des zones LZ()
. les types de zones TZ() (1,2,3,4)

Z$()	LZ()	TZ()
NOM:	15	1
PRENOM:	12	1
TELEPH:	20	1
VILLE:	15	1
CPOST:	4	3
Noms des zones	longueur des zones	type des zones

ZN$(1)	ZN$(2)	ZN$(3)	
FIDON	JEANNE	355-22-56	

Zones definies dans FIELD#

Pour une saisie plus élaborée 'caractère par caractère', cf. programme EDIR.

```
10 ' SAISR 23.5.80
20 '
30 '      SAISIE POUR FICHIER RANDOM
40 '      --------------------------
50 '
60 '    Permet, en cours de saisie,de revenir sur des zones arrieres
70 '
80 '    TZ() : 1   Chaines
90 '           2   Entiers   -32000<X<32000
100 '          3   Simple precision
110 '          4   X<256
120 '
140 '    LZ()  : Longueur des zones
150 '    NZ$() : Noms des zones
160 '
170 OPEN "R",1,"SAIS"
180 NZ=5                                                  ' Nombre de zones
190 DATA "NOM:"     ,15,1
200 DATA "PRENOM:"  ,12,1
210 DATA "TELEPH:"  ,20,1
220 DATA "VILLE:"   ,15,1
230 DATA "CPOST:"   ,4 ,3
240 '
250 FOR I=1 TO NZ:READ NZ$(I):READ LZ(I):READ TZ(I):NEXT I ' Nom,Longueur,Type
260 '
270 FIELD #1,LZ(1) AS ZN$(1),LZ(2) AS ZN$(2),LZ(3) AS ZN$(3),LZ(4) AS ZN$(4),L
Z(5) AS ZN$(5)
280 FIELD #1,LZ(1) AS NOM$,LZ(2) AS PRENOM$
290 '
300 PRINT:INPUT "NO ENREG? ";NE
310 PRINT:GET #1,NE:GOSUB 350:PUT #1,NE
320 GOTO 300
330 '---------------------------------------------------------------
340 '                                                          SAISIE
350 PRINT:PRINT  "      R : retour zone arriere":PRINT
360 FOR I=1 TO NZ
370   PRINT NZ$(I);TAB(20);
380   ON TZ(I) GOSUB 460,470,480,490           ' Affichage ancienne valeur
390   PRINT TAB(45); :X$="": INPUT X$
400   IF X$="R" THEN IF I>1 THEN I=I-1:GOTO 370 ELSE GOTO 370
410   IF X$="" GOTO 430
420   ON TZ(I) GOSUB 510,520,530,540
430 NEXT I
440 RETURN
450 '
460 PRINT ZN$(I);:RETURN                  ' TZ=1  Affichage ancienne valeur
470 PRINT CVI(ZN$(I));:RETURN             ' TZ=2
480 PRINT CVS(ZN$(I));:RETURN             ' TZ=3
490 PRINT ASC(ZN$(I));:RETURN             ' TZ=4
500 '
510 LSET ZN$(I)=X$:RETURN                 ' TZ=1  Nouvelle valeur
520 LSET ZN$(I)=MKI$(VAL(X$)):RETURN
530 LSET ZN$(I)=MKS$(VAL(X$)):RETURN
540 LSET ZN$(I)=CHR$(VAL(X$)):RETURN
550 '
560 'run
570 '
580 'NOM:       DUPONT
590 'PRENOM:    JEAN
600 'TELEPHONE: 679-99-88  700-99-00
610 'VILLE:     PARIS
620 'CPOST:     75009
```

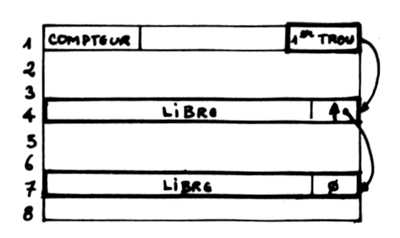
ALLOCATION D'ENREGISTREMENTS POUR FICHIER RANDOM MICROSOFT 5.

La fonction LOF(x) en Microsoft 5. n'indique plus la fin de fichier comme c'était le cas avant l'apparition de CPM. Cette fonction peut être remplacée par la gestion d'un compteur dans l'enregistrement numéro 1 du fichier. La récupération des enregistrements devenus libres peut se faire à l'aide d'une bit-map (cf. Basic et ses Fichiers, tome 1, page 85).

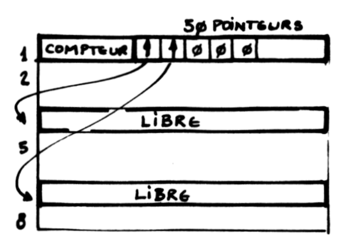
Une autre façon de procéder consiste à chaîner les enregistrements inutilisés entre eux. Chaque nouvel enregistrement supprimé est mis en tête de chaîne par exemple. Ce système oblige à réserver deux octets systématiquement dans chaque enregistrement pour le chaînage.

Pour se prémunir contre d'éventuels incidents, il faut penser à repérer les enregistrements libres par un code (code ASCII nul par exemple) de façon à être capable de régénérer la chaîne.

Bien que la gestion d'un chaînage soit relativement simple, nous proposons une méthode plus facile à programmer, applicable seulement si le nombre d'enregistrements libres peut être estimé. On prévoit dans l'enregistrement numéro 1 du fichier, 50 pointeurs vers les enregistrements supprimés.



* CHAINAGE DES 'TROUS'



* POINTEURS VERS LES ENREGISTREMENTS LIBRES

```
10 ' MIC5        ALLOCATION D'ENREGISTREMENTS POUR FICHIER RANDOM MICROSOFT 5.
20 '
30 '         La version 5. Microsoft ne dispose pas de la fonction LOF(x)
40 '         Un compteur dans l'enregistrement no 1 la remplace.
50 '         Les adresses des enregistrements supprimes sont stockees dans enreg 1
70 '
80 OPEN "R",#1,"MIC5"
90 FIELD #1,15 AS NOM$
100 FIELD #1,2 AS CPT$                    ' Compteur
105 FIELD #1,128 AS I1$
110 DIM VD$(50)                           ' Pointeurs enreg vides
120 FOR I=1 TO 50                         ' Pointeurs enregistrements libres
130   FIELD #1,2+2*(I-1) AS D$,2 AS VD$(I)
140 NEXT I
150 '---------------------------- Initialisation enregistrement no 1 avec 0 ASCII
160 IF LOF(1)=0 THEN LSET I1$=STRING$(CHR$(0),128):PUT #1,1
170 '--------
180 GET #1,1:CPT=CVI(CPT$):IF CPT<2 THEN CPT=2
190 '
200 FOR CPT=CPT TO 1000                   ' MAJ Eventuelle du compteur
210  GET #1,CPT
220  IF ASC(NOM$)=0 THEN PRINT "COMPTEUR=";CPT:GOTO  260
230 NEXT CPT
240 PRINT "FICHIER PLEIN":STOP
250 '-------------------------------- Menu
260 INPUT "Mode? ";M$
270 IF M$="C" THEN GOSUB 310             ' Creation
280 IF M$="S" THEN GOSUB 380             ' Suppression
290 GOTO 260
300 '================================================ CREATION
310 GOSUB 460                            ' Appel recherche enregistrement
320 X$="":INPUT "Nom? ";X$:IF X$="" THEN RETURN
330 LSET NOM$=X$:PUT #1,NE
340 PRINT "Range en:";NE
350 GOSUB 540                            ' Appel MAJ compteur ou pointeurs
360 GOTO 310
370 '================================================ SUPPRESSION
380 INPUT "Quel enregistrement? ";NE:IF NE=1 OR NE>=CPT THEN PRINT "Erreur":RETURN
400 GET #1,NE
410 IF ASC(NOM$)=0 THEN PRINT "Deja vide":RETURN
420 LSET NOM$=CHR$(0):PUT #1,NE:GOSUB 580
430 '  Pour 0 ASCII enreg complet faire:105 FIELD #1,128 AS I1$
435 '                                  420 LSET I1$=STRING$(128,CHR$(0)):PUT #1,NE
440 RETURN
450 '------------------------------- Recherche enregistrement libre
460 GET #1,1
470 FOR PE=1 TO 50
480   IF CVI(VD$(PE))=0 THEN 500
490   NE=CVI(VD$(PE)):GET #1,NE:IF ASC(NOM$)=0 THEN RETURN ELSE PRINT "Erreur point
eur":GOTO 500
500 NEXT PE
510 PE=0:NE=CPT                         ' Allocation en fin de fichier
520 RETURN
530 '---------------------------------------- MAJ compteur ou pointeurs enr libres
540 GET #1,1
550 IF PE=0 THEN CPT=CPT+1:LSET CPT$=MKI$(CPT)::PUT #1,1:RETURN
560 LSET VD$(PE)=MKI$(0):PUT #1,1:RETURN
570 '---------------------------------------- MAJ pointeurs pour suppression
580 GET #1,1
590 FOR PE=1 TO 50
600   IF CVI(VD$(PE))=0 THEN LSET VD$(PE)=MKI$(NE):PUT #1,1:RETURN
610 NEXT PE
620 PRINT "Y'a plus de place pour les pointeurs":STOP
630 '-----------------------------------------------------
```

```
10 ' CHAI5      CHAINAGE D'ENREGISTREMENTS INUTILISES POUR FICHIER RANDOM
20 '
30 '         La version 5. Microsoft ne dispose pas de la fonction LOF(x)
35 '         Un compteur dans l'enregistrement no 1 la remplace
40 '         Les enregistrements supprimes sont chaines entre eux
60 '
80 OPEN "R",#1,"chai5"
90 FIELD #1,15 AS NOM$
100 FIELD #1,2 AS CPT$                      ' Compteur
105 FIELD #1,128 AS I1$
110 DIM VD$(50)
120 FIELD #1,126 AS D$,2 AS TROU$
122 '-------------------------------------- Initialisation enreg no 1 avec 0 ASCII
124 IF LOF(1)=0 THEN LSET I1$=STRING$(CHR$(0),128):PUT #1,1
125 '-------
150 GET #1,1:CPT=CVI(CPT$):IF CPT<2 THEN CPT=2
155 '
160 FOR CPT=CPT TO 1000                     ' MAJ Eventuelle du compteur
170  GET #1,CPT
180  IF ASC(NOM$)=0 THEN PRINT "COMPTEUR=";CPT:GOTO 220
190 NEXT CPT
200 PRINT "FICHIER PLEIN":STOP
210 '-------------------------------- Menu
220 INPUT "Mode? ";M$
230 IF M$="C" THEN GOSUB 290               ' Creation
240 IF M$="S" THEN GOSUB 390               ' Suppression
250 GOTO 220
280 '============================================ CREATION
290 GOSUB 470                              ' Appel recherche enregistrement
300 INPUT "Nom? ";X$
310 IF X$="" THEN RETURN
315 GET #1,NE
320 LSET NOM$=X$
330 PUT #1,NE
340 PRINT "Range en:";NE
350 GOSUB 560                              ' Appel MAJ compteur ou pointeurs
360 GOTO 290
380 '============================================ SUPPRESSION
390 INPUT "Quel enregistrement? ";NE:IF NE=1 OR NE=>CPT THEN PRINT "Erreur":RETURN
400 GET #1,NE
405 IF ASC(NOM$)=0 THEN PRINT "Deja vide":RETURN
406 PRINT NOM$
410 LSET NOM$=CHR$(0)
420 PUT #1,NE
430 GOSUB 600
440 RETURN
460 '------------------------------- Recherche enregistrement libre
470 GET #1,1
480 IF CVI(TROU$)<>0 THEN NE=CVI(TROU$):GET #1,NE:ATROU$=TROU$:PE=1:RETURN
520 PE=0:NE=CPT                           ' Allocation en fin de fichier
530 RETURN
550 '--------------------------------------- MAJ compteur ou pointeurs enr libres
560 GET #1,1
570 IF PE=0 THEN CPT=CPT+1:LSET CPT$=MKI$(CPT):PUT #1,1:RETURN
580 GET #1,1:LSET TROU$=ATROU$:PUT #1,1:RETURN
590 '--------------------------------------- MAJ pointeurs pour suppression
600 GET #1,1
610 X$=TROU$
615 LSET TROU$=MKI$(NE):PUT #1,1
620 GET #1,NE
625 LSET TROU$=X$
630 PUT #1,NE
635 RETURN
650 '-----------------------------------------------
```

CHAPITRE 2
ACCES PAR CLE

Un opérateur n'a pas à connaître les numéros d'enregistrements où sont rangés des clients, des produits, des factures, etc... Comment donc retrouver un enregistrement contenant une clé cherchée ?

1/ Par recherche séquentielle

2/ Par table d'index

3/ Par HASH-CODE

RECHERCHE SEQUENTIELLE

La façon la plus simple, (mais aussi la moins rapide), pour retrouver une clé dans un fichier consiste à explorer séquentiellement celui-ci jusqu'à ce que la clé cherchée soit retrouvée.

```
 10 OPEN "R",#1,"STCK"
 20 FIELD #1,12 AS REF$,25 AS LIB$,....
 30 '
100 INPUT "Cle? ";CLE$          ' Entrer seulement les premieres lettres
105 L=LEN(CLE$)
110'
120 FOR I=1 TO LOF(1)           ' Lire tout le fichier
130   GET #1,I
140   IF CLE$=LEFT$(REF$,L) THEN GOTO 200
150 NEXT I
160'
170 PRINT "La cle cherchee n'existe pas":goto 100
180'
200 PRINT REF$,LIB$             ' La cle est trouvee
210 GOTO 100
```

Remarque importante : la zone REF∅ étant complétée par des espaces à droite au moment de l'écriture dans le fichier, (voir LSET), nous comparons CLE∅ à LEFT∅(REF∅, LEN(CLE∅)). En outre, ceci permet à l'opérateur, pour la recherche, de n'entrer que les premières lettres du nom.

```
-REF$    LIB$
--------------------------------
!R5    !xxxxxxxxxxxxxxxxxxxxxxxx!
!CX    !xxxxxxxxxxxxxxxxxxxxxxxx!
!      !xxxxxxxxxxxxxxxxxxxxxxxx!
--------------------------------
```

RECHERCHE PAR TABLE D'INDEX

L'exploration d'une table d'index en mémoire centrale est bien sûr plus rapide que l'exploration d'un fichier.

LECTURE DU FICHIER EN DEBUT DE SESSION

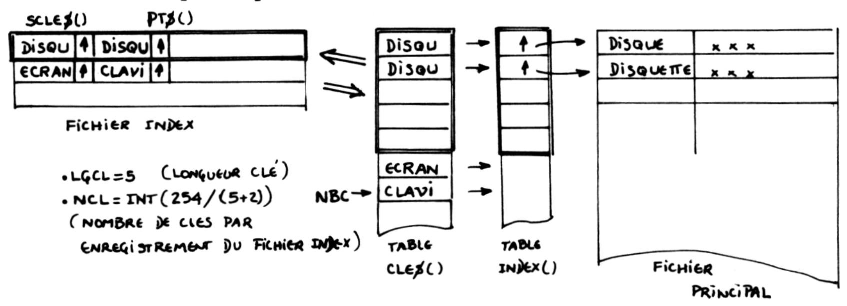
Une table d'index est créée au début de chaque session de travail en lisant tout le fichier (cf. Le Basic et ses Fichiers, Tome 1, p. 95).

INDEX SAUVEGARDE SUR DISQUE

Afin d'économiser le temps de constitution de la table d'index, celle-ci est généralement sauvegardée sur disque dans le fichier principal ou un fichier indépendant.

Compte tenu de la place occupée en mémoire centrale par l'index, seules les premières lettres des clés peuvent y être gardées. Bien entendu, dans ce cas, plusieurs accès disques sont nécessaires pour retrouver une clé si plusieurs clés ont les mêmes premières lettres.

Sur TRS-80, un enregistrement de 256 caractères peut contenir 256/(5+2)=36 clés, si les clés ont une longueur de 5 caractères et que 2 caractères sont nécessaires pour les pointeurs vers le fichier principal.



La recherche d'une clé se programme ainsi :

NBC : Nombre de clés dans la table d'index
LGCL : Longueur des clés dans la table d'index (LGCL=5)

```
500 INPUT "Cle cherchee? ";X$        ' Entrer seulement les premieres lettres
505 L=LEN(X$)
510 '
520 FOR I=1 TO NBC                   ' Lecture de la table CLE$()
530   IF LEFT$(CLE$(I),L)<>X$ AND LEFT$(X$,LGCL)<>CLE$(I) THEN GOTO 570
540   GET #1,INDEX(I)
550   '
560   IF X$=LEFT$(REF$,L) THEN PRINT REF$;LIB$:GOTO 500  ' Cle trouvee
570 NEXT I
580 '
590 INPUT "Nouvelle cle OK (O/N)? ";R$:IF R$<>"O" THEN GOTO 500
595 RANG=LOF(1)+1                    ' Rgmt nouveau produit en fin de fichier
600 LSET REF$=X$
605 '  Saisie article
610 '
620 PUT #1,RANG
630 NBC=NBC+1:CLE$(NBC)=X$:INDEX(NBC)=RANG    ' Ajout cle en fin de table d'index
640 GOSUB 1000                                ' Sauvegarde de la table d'index
650 GOTO 500
```

Sauvegarde de la table d'index :

En cas d'incident (coupure de tension par exemple), la table d'index en mémoire centrale est perdue. Aussi convient-il de la sauvegarder sur disque à chaque ajout de clé, du moins la partie modifiée.

L'ajout de clé se faisant en fin de table, le numéro de la case modifiée est NBC.

NCL :	nombre de clés par enregistrement du fichier index (256/(5+2)=36)
SCLE$() :	définie dans FIELD# du fichier index pour sauvegarde de CLE$() (contient NCL éléments)
PT$() :	définie dans FIELD# du fichier index pour sauvegarde de INDEX()
NBC :	nombre de clés dans la table index

```
  10 OPEN "R",#1,"STCK"
  20 FIELD #1,12 AS REF$,25 AS LIB$,...
  30 OPEN "R",#2,"INDEX"              ' Fichier index
  35 LGCL=5:NCL=36                    ' Longueur des cles
  40 FOR I=1 TO NCL
  50   FIELD #2,(LGCL+2)*(I-1) AS D$,(LGCL) AS SCLE$(I),2 AS PT$(I)
  60 NEXT I
  70 '
1000 DB=INT((NBC-1)/NCL)              ' DB:No de bloc a sauvegarder
1010 K=DB*NCL+1                       ' K :Debut de la table a sauvegarder
1015 GET #2,DB+1
1020 FOR J=1 TO NCL
1030   IF CLE$(K)="" THEN PUT #2,DB+1:RETURN
1040   LSET SCLE$(J)=CLE$(K):LSET PT$(J)=MKI$(K):K=K+1
1050 NEXT J
1060 PUT #2,DB+1
1070 RETURN
```

Remarque : l'instruction 1015 est nécessaire. En effet, sans celle-ci, la mémoire tampon contiendrait les valeurs d'un ancien bloc.

La totalité de l'index peut également être sauvegardée seulement en fin de session. Mais si celle-ci est interrompue, l'index ne sera pas à jour pour les sessions ultérieures.

Transfert de la sauvegarde sur disque en mémoire centrale :

En début de session, les tables CLE$() et INDEX() sauvegardées sur disque doivent être amenées en mémoire centrale :

```
100 NBC=0
110 FOR I=1 TO 10                    ' 10 enregistrements pour le fichier index
120   GET #2,I
130   FOR J=1 TO NCL
140     IF ASC(SCLE$(J))=0 THEN 170
150     NBC=NBC+1:CLE$(NBC)=SCLE$(J):INDEX(NBC)=CVI(PT$(J))
160   NEXT J
170 NEXT I
```

Régénération d'un index

Il doit toujours être prévu un mode capable de regénérer l'index en cas de destruction de celui-ci ou d'incohérence avec le fichier principal. Pour cela, il suffit de régénérer la table d'index par une lecture séquentielle du fichier, puis de la sauvegarder.

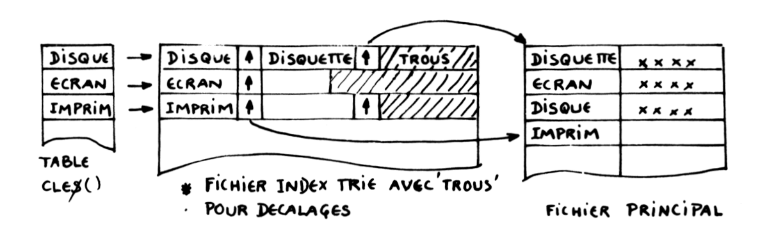
```
200 IF M$="CRI"  THEN GOSUB 2000:STOP                ' Mode creation index


2000 CLOSE #2:KILL "INDEX":OPEN "R",#2,"INDEX"      ' RAZ du fichier index
2010 FOR I=1 TO 500:CLE$(I)="":INDEX(I)=0:NEXT I   ' RAZ de CLE$() et INDEX()
2015 NBC=0
2020 FOR I=1 TO LOF(1)               ' Lecture du fichier principal
2030   GET #1,I
2040   IF ASC(REF$)=0 THEN  2060                    ' Enregistrement vide?
2050   NBC=NBC+1:CLE$(NBC)=REF$:INDEX(NBC)=I        ' Ajout de la cle
2060 NEXT I
2070 '----------- Sauvegarde de la table  CLE$() dans le fichier index.
2100 W=1
2110 FOR I=1 TO 10                   ' 10 enreg maxi pour l'index
2120   GET #2,I
2130   FOR J=1 TO NCL                ' NCL:nb de cles par enreg du fichier index
2140     IF CLE$(W)="" THEN PUT #2,I:RETURN
2150     LSET SCLE$(J)=CLE$(W):LSET PT$(J)=MKI$(INDEX(I)):W=W+1
2160   NEXT J
2170   PUT #2,I
2180 NEXT I
2190 RETURN
```

INDEX AVEC 'TROUS'

Pour des fichiers 'stables', c'est-à-dire, avec peu d'ajouts de clés, des trous dispersés dans un index trié évitent des décalages nombreux en cas d'insertion de clé.

Toutefois, si les ajouts de clés viennent à l'emporter sur les suppressions, une réorganisation de l'index est nécessaire.



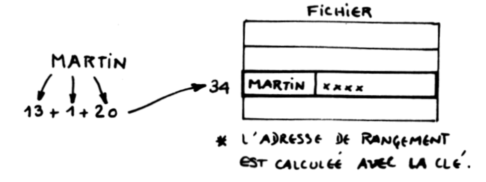
Exemple : un dictionnaire pourrait être construit sur ce principe :

table — FICHIER DICTIONNAIRE(TRIE)

table		FICHIER DICTIONNAIRE(TRIE)				
ABAQUE	-->	ABAQUE	xxxxxx	ABATTOIR	xxxxxxx	place libre pour insertions
BAGAGE	-->	BAGAGE	xxxxxx	BARRE	xxxxxxx	
	-->					

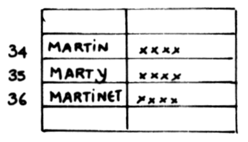
HASH-CODE

Bien que peu connu, le principe de cette méthode d'accès est simple. L'adresse de rangement d'une clé est définie par un calcul effectué sur celle-ci (la somme des positions dans l'alphabet des premières lettres par exemple).



Le même calcul étant effectué à la lecture, nous retrouvons une clé cherchée en un seul accès disque.

Bien entendu, plusieurs clés fournissent la même adresse de rangement. Dans ce cas, les 'collisions' peuvent être résolues en rangeant la nouvelle clé à côté de l'adresse calculée ou par des chaînages vers une zone des collisions.



Plus le taux d'occupation du fichier est grand, plus les collisions augmentent (et le temps d'accès aussi bien sûr). On doit donc consentir une perte de place de l'ordre de 30 à 40 %.

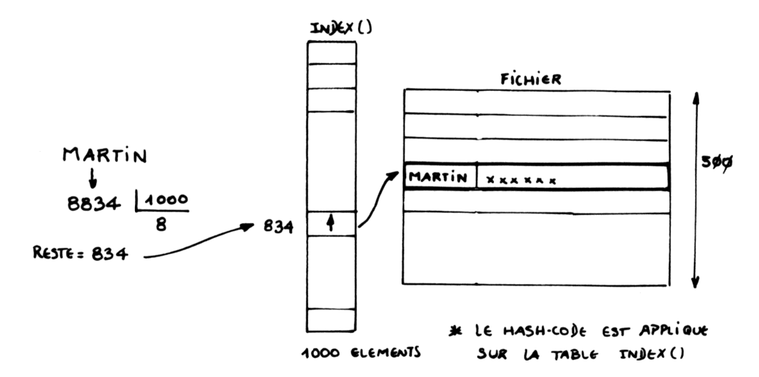
L'algorithme de rangement proposé n'est évidemment qu'un exemple. En réalité, un bon algorithme doit répartir les clés le plus uniformément possible dans le fichier.

Appliquée sur des **fichiers virtuels** (c'est-à-dire, où la place disque n'est allouée que pour les enregistrements où il y a écriture), cette méthode est très pratique. Hélas, les DOS actuels ne font l'allocation que par blocs de 1K minimum.

HASH-CODE SUR UNE TABLE

Si la mémoire centrale disponible est suffisante, le système de Hash-Code peut être appliqué sur une table (sur l'exemple, les nouvelles clés sont rangées en fin de fichier).

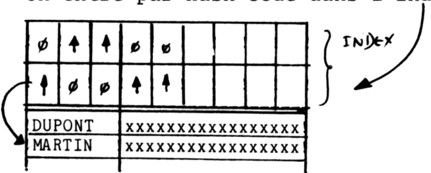
La table d'index doit bien entendu être sauvegardée sur disque.



HASH-INDEX

La table en mémoire centrale ci-dessus peut être supprimée. L'accès par Hash-Code se fait alors sur un index de pointeurs résident sur disque. Deux accès disques sont nécessaires pour accéder à l'enregistrement cherché.

On entre par Hash-Code dans l'index de pointeurs.

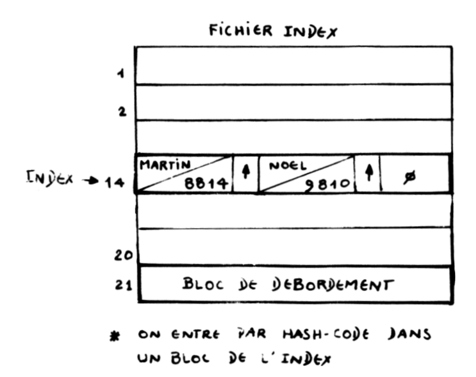


Une variante de la méthode précédente consiste à définir des 'blocs' d'index dans lesquels on 'entre' par Hash-Code. Puis, à l'intérieur d'un bloc, la recherche se fait séquentiellement.

Un bloc de débordement est prévu pour les clés qui n'auraient pas trouvé de place dans le bloc à l'adresse calculée.

```
INDEX=CLE MOD 20              ' 20 blocs dans l'index
```

A l'intérieur de l'index, nous rangeons soit la clé alphabétique, soit la clé numérique (calculée par Hash-Code), plus économique en place.



HASH-CODE ET ALLOCATION DYNAMIQUE

Nous proposons une variante de Hash-Code économique en place mémoire et assurant l'allocation dynamique des enregistrements.

Ajout d'une clé :

Nous faisons correspondre à une clé alphabétique, par un algorithme quelconque, une clé numérique que nous rangeons dans la première case libre d'une table HASH%() puis nous rangeons la nouvelle clé dans l'enregistrement correspondant.

Recherche d'une clé :

On calcule, comme pour la création, une clé numérique dont on recherche la position dans la table HASH%. Il suffit ensuite de lire l'enregistrement correspondant. Si la même clé numérique existe en plusieurs exemplaires, les enregistrements correspondants doivent être lus jusqu'à ce que la clé cherchée soit trouvée.

Suppression d'une clé :

En cas de suppression d'enregistrement, la case associée de HASH%() se trouve libérée pour un ajout ultérieur. Ainsi, l'allocation dynamique des enregistrements est assurée implicitement.

Nous avons choisi de repérer les enregistrements inutilisés par -32000, 0 repérant la fin de table. En choisissant 0 pour identifier les enregistrements libres, il nous faudrait explorer systématiquement toute la table lors d'une recherche de clé.

La table HASH%() occupe N*2 octets, ce qui est moins encombrant qu'une table d'index classique. La recherche séquentielle, pénalisante en interprété, devrait être transparente en compilé.

Si on choisit de trier la table HASH%() afin d'y faire une recherche dichotomique, il devient nécessaire de lui associer une table d'index (cf. accès indexé et allocation dynamique).

FICHIER PRINCIPAL

MARTIN

1ère LETTRE X26 X26 + 2ème LETTRE X 26 + 3ème LETTRE = 8834

MARTINET 8834

HASH%()

1	8834
2	4421
3	5230
4	8834

CLES 1-32000

1	SAUVEGARDE DE HASH%()	
2		
3		
4	MARTIN	XXXX
5	DUPONT	XXXX
6	BOE	XXXX
7	MARTINET	XXXX

* HASH-CODE ET ALLOCATION DYNAMIQUE

```
10 ' HASH 18.11.80
30 '                    HASH-CODE et ALLOCATION DYNAMIQUE
40 '
60 ' Les enregistrements 1 a 3 servent a la sauvegarde de HASH%()
70 '-------------
80 OPEN "R",#1,"HAS"
90 EF$=CHR$(31)
460 '--------------------------------------------
470 NCLES=64          ' A ADAPTER  (125 SUR TRS80)  :nbre de cles par enregistrement
490 DIM HASH%(NCLES*3)              ' Table des cles numeriques
500 DIM SHASH$(NCLES)               ' Definie dans FIELD# pour sauvegarde de HASH%
510 FIELD #1,12 AS N$
515 FIELD #1,(2*NCLES) AS I1$       ' Buffer complet de #1
530 FOR I=1 TO NCLES:FIELD #1,2*(I-1) AS D$,2 AS SHASH$(I):NEXT I
540 GOSUB 1120                      ' Appel lecture HASH%
550 '----------
560 INPUT "Mode ? C,A ";M$
570 IF M$="C" THEN GOSUB 610        ' Appel CREATION
580 IF M$="A" THEN GOSUB 1220       ' Appel SUPPRESSION
590 GOTO 560
600 '========================================== CREATION/RECHERCHE D'UNE CLE
610 PRINT:INPUT "NOM? ";NOM$
620 IF LEN(NOM$)<3 THEN RETURN
630 GOSUB 750                           ' Appel recherche cle
640 ON R GOTO 650,670
650 PRINT N$,RANG:GOTO 610              ' La cle existe
660 '
670 INPUT "NOUVEAU NOM OK? ";R$:IF R$<>"O" THEN GOTO 610  ' La cle n'existe pas
680 PRINT EF$
690 LSET I1$=STRING$(CHR$(0),NCLES*2)   ' Initialisation buffer avec 0 ASCII
700 LSET N$=NOM$:PUT #1,RANG:HASH%(RANG)=CLE:GOSUB 930:GOTO 610
710 '---------------------------------- Recherche d'une cle
720 '
730 '          Entree:NOM$   Retour:   RANG: Adresse de rangement dans le fichier
740 '                                  R=1  :  La cle existe /R=2 : N'existe pas
750 FOR I=1 TO 3:X(I)=ASC(MID$(NOM$,I,1))-64:NEXT I
760 CLE=X(1)*26*26+X(2)*26+X(3)           ' Calcul d'une cle numerique
770 PLIB=0                                ' Position libre dans le fichier
780 FOR I%=3+1 TO 3*NCLES                 ' Lecture de la table HASH%
790   IF HASH%(I%)=0 THEN 880             ' Fin de table?
800   IF HASH%(I%)<>CLE THEN 840
810   GET #1,I%
820   IF NOM$=LEFT$(N$,LEN(NOM$)) THEN RANG=I%:R=1:RETURN   ' Nom trouve
830 '
```

```
840   IF PLIB=0 THEN IF HASH%(I%)=-32000 THEN PLIB=I%        ' -32000:case libre
850 NEXT I%
860 PRINT "C'est plein":STOP
870 '
880 IF PLIB=0 THEN PLIB=I%
890 R=2:RANG=PLIB:RETURN
900 '----------------------------------- Sauvegarde de la table HASH%()
910 '
920 '          DB : No du bloc de HASH%() a sauvegarder.(0,1,2,..)
930 DB=INT((RANG-1)/NCLES)
940 NB=DB*NCLES
950 GET #1,DB+1
960 FOR J=1 TO NCLES
970     NB=NB+1:LSET SHASH$(J)=MKI$(HASH%(NB))
980 NEXT J
990 PUT #1,DB+1
1000 RETURN
1010 '------------------------------------------------------------
1050 '
1060 '
1070 '
1080 '
1090 ' L'ecriture directe dans le quatrieme enreg cree les 3 premiers avec
1095 ' des valeurs quelconques .
1100 '
1110 '                          LECTURE DE LA TABLE HASH%()
1120 NB=0
1125 IF LOF(1)=0  THEN LSET I1$=STRING$(CHR$(0),NCLES*2):FOR I=1 TO 3: PUT #1,I:
   NEXT I:LSET SHASH$(1)=MKI$(32000):PUT #1,1  ' Initialisation index avec 0 ASCII
1130 FOR I=1 TO 3
1140   GET #1,I
1150   FOR J=1 TO NCLES
1160     NB=NB+1:HASH%(NB)=CVI(SHASH$(J))
1170   NEXT J
1180 NEXT I
1190 RETURN
1200 '=========================================== SUPPRESION
1220 INPUT "Nom? ";NOM$
1230 IF LEN(NOM$)<3 THEN RETURN
1240 GOSUB 750:ON R GOTO 1260,1250          ' Appel recherche cle
1250 PRINT:PRINT "N'existe pas":GOTO 1220
1260 LSET I1$=STRING$(CHR$(0),NCLES*2):PUT #1,RANG      ' Suppression de l'enregist
rement
1270 HASH%(RANG)=-32000                     ' -32000 pour reperer une case vide
1280 GOSUB 930
1290 GOTO 1220
1292 '-------------------------------------------------------------
1293 '   ASSOCIER CE PROGRAMME AVEC 'SAIZ' pour obtenir un programme de saisie
1294 '   avec acces par cle.
1295 '   1/ 'MERGER' les 2 programmes sauvegardes en ASCII par 'SAVE "XX",A"'
1296 '   2/ ajouter  690 LSET I1$=...:GOSUB 1320
1297 '   3/ ajouter  650 PRINT EF$:GOSUB 1320:PUT #1,RANG:GOTO 610
1298 '   4/ Supprimer 86 et 445
```

Régénération de la table HASH%()

Un mode de régénération de la table HASH%() (en cas de destruction) peut se programmer ainsi :

```
2000  LSET I1$=STRING$(CHR$(0),NCLES*2):FOR I=1 TO 3:PUT #1,I:NEXT I
2010  FOR I=1 TO NCLES*3:HASH%(I)=0:NEXT I    ' RAZ de la table HASH%()
2020  '
2030  FOR NE=1 TO LOF(1)              ' Lecture du fichier(cf EOF pour 5.)
2040     GET #1,NE
2050     IF ASC(N$)=0 THEN CLE=-32000:GOTO 2100         ' Enreg vide?
2060     FOR I=1 TO 3:X(I)=ASC(MID$(N$,I,1))-64:NEXT I  ' Calcul de clé
2070     CLE=X(1)*26*26+X(2)*26+X(3)
2090 '
2100     HASH%(NE)=CLE:RANG=NE
2110     GOSUB 930                              ' Appel sauvegarde de HASH%()
2120   NEXT NE
```

Remarques sur le programme :

Un sous-programme de recherche d'une clé fournit en retour un indicateur R=1 si la clé existe ou R=2 si la clé n'existe pas. A chaque ajout, le morceau de HASH%() modifié est sauvegardé sur disque.

Remarques sur le Hash-Code classique : alors que l'accès indexé, lorsque l'index est trié, permet d'obtenir une liste des clés dans l'ordre instantanément, le Hash-Code ne le permet pas.

Si une adresse de rangement, dans le but d'obtenir une meilleure répartition à l'intérieur du fichier, a été calculée sur toutes les lettres d'un nom, il n'est pas possible de retrouver un nom avec seulement les premières lettres.

En revanche, le système proposé permet, compte tenu des poids affectés aux premières lettres, de retrouver un ensemble de clés dans un certain voisinage.

Exemple : les noms commençant par 'MA' seront retrouvés par :

```
100 INPUT "Nom? ";X$
110 '
120 calcul de cle
130 '
200 FOR I=1 TO 100
210   IF CLE<HASH%(I)-100 OR CLE>HASH(I)+100 THEN GOTO 300
220   GET #1,I
230   PRINT N$
300 NEXT I

RUN
       Cle? MA

       MARTIN
       MARTY
       MARTINET
```

Recherche rapide dans la table HASH :

La recherche séquentielle dans la table 'HASH' pourrait être accélérée en compactants ses éléments par MKI$, (elle deviendrait alors une table de chaînes), et en y effectuant la recherche de clé à l'aide de INSTR.

```
10 '       RECHERCHE et AJOUT dans une TABLE avec 'INSTR'
20 '
30 '
40 '     Une recherche sequentielle dans une table en BASIC interprete est longue.
50 '     Le compactage de ses elements sous forme d'une table de chaines
60 '     permet d'effectuer plus rapidement la recherche a l'aide de la
70 '     fonction 'INSTR'
80 '
90 '     La table HASH$() de dimension 3 est sauvegardee dans un fichier RANDOM.
100 '
110 '
120 NCLES=10                    ' Nombre de cles par chaine (122 maxi)
130 LCHAI=NCLES*2
140 OPEN "R",#1,"HAS"           ' Sauvegarde de la table HASH$()
150 FIELD #1,(LCHAI) AS I1$
154 '------ Initialisation des 3 premiers enreg avec 0 ASCII
155 IF LOF(1)=0 THEN LSET I1$=STRING$(CHR$(0),NCLES*2):FOR I=1 TO 3:PUT #1,I:NEXT I

160 '---------------------------- Lecture de la table HASH$() sauvegardee
170 FOR I=1 TO 3
180  GET #1,I
190  HASH$(I)=I1$
200 NEXT I
210 '-------------- Recherche
220 '
230 INPUT "Quel nombre cherchez vous? ";X
240 X$=MKI$(X)
250 '
260 FOR LI=1 TO 3               ' 3 chaines de NCLES
270  DR=1                       ' Debut de recherche
280 '
290  P=INSTR(DR,HASH$(LI),X$):IF P=0 THEN 340
300  IF (P MOD 2)=0 THEN DR=P+1:GOTO 290
310  PRINT "Position de X:";P;LI;CVI(MID$(HASH$(LI),P,2))
320  GOTO 230
330 ' --------- Recherche fin de table
340  DR=1
350  PZ=INSTR(DR,HASH$(LI),MKI$(0)):IF PZ=0 THEN 380
360  IF (PZ MOD 2)=0 THEN DR=DR+1:GOTO 350
370  GOTO 410
380 NEXT LI
390 STOP
400 '------- Ajout en fin de table
410 X$=HASH$(LI):HASH$(LI)=LEFT$(X$,PZ-1)+MKI$(X)+RIGHT$(X$,LCHAI-PZ-1)
420 LSET I1$=HASH$(LI):PUT #1,LI
430 PRINT "Element insere"
440 PRINT "Position:";PZ
450 GOTO 230
```

CHAPITRE 3

LES TRIS

Tout a été dit. De nombreux livres y sont consacrés. Et pourtant, devant toutes ces méthodes de tri, beaucoup hésitent. Quelle méthode choisir ?

Nous ne reprendrons pas tous les aspects théoriques sur les tris. Nous en ferons simplement un rapide panorama.

En fonction de quoi choisir son tri ?

- du nombre d'éléments à trier,
- éventuellement de l'état de la liste à trier (liste presqu'en ordre).

Intéressons-nous d'abord aux deux tris les plus simples, les plus connus, mais aussi les moins rapides : **Bubble** et **Ripple**.

RIPPLE

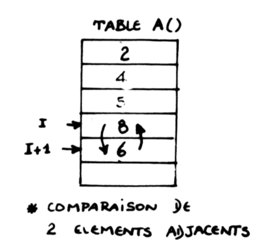
Les éléments adjacents de la table à trier sont successivement comparés et inversés s'ils ne sont pas dans l'ordre. Les comparaisons se font en progressant de 1. Lorsque toute la table a été explorée, le plus grand élément doit être en fin de table.

Une seconde exploration de la table permet d'amener le plus grand des N-1 éléments restants en avant dernière position de la table.

Après N passages au maximum, la table sera en ordre.

En réalité, on positionne généralement un témoin d'inversion à 1 pour se souvenir en fin de table s'il y a eu ou non inversion, car si ce n'est pas le cas, on peut en conclure que la table est en ordre et stopper le tri avant N passages.

```
95  K=N                    ' Tri RIPPLE
97'
100 INV=0
110 FOR I=1 TO K-1
120   IF A(I+1)<A(I) THEN SWAP A(I+1),A(I):INV=1
130 NEXT I
140 IF INV<>0 THEN K=K-1:GOTO 100    ' On diminue de 1 le nombre d'elements a explorer
                                     ' A chaque passage.
```



* COMPARAISON DE 2 ÉLÉMENTS ADJACENTS

Si le tri doit s'effectuer sur une table déjà en ordre avec quelques éléments ajoutés en fin de table, le tri devra être fait en sens inverse. Ainsi X passages seulement seront nécessaires pour insérer les X éléments ajoutés.

```
95 K=1
97 '
100 INV=0
110 FOR I=N-1 TO K STEP-1
120   IF A(I+1)<A(I) THEN SWAP A(I+1),A(I):INV=1
130 NEXT I
140 IF INV<>0 THEN K=K+1:GOTO 100
```

3
5
7
8
10
2
9
4

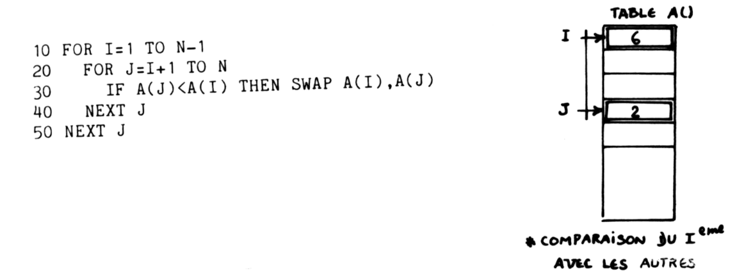
DÉJÀ EN ORDRE (3 à 10) — X ÉLÉMENTS AJOUTÉS (2, 9, 4)

* AJOÛTS DANS UNE TABLE TRIÉE

BUBBLE

On compare d'abord le premier élément aux N-1 autres en inversant, chaque fois que l'un d'eux est plus petit, de façon à amener le plus petit en première position.

En procédant ainsi sur les N-1 éléments restants, on amène le plus petit de ceux-ci en seconde position dans la table, etc.



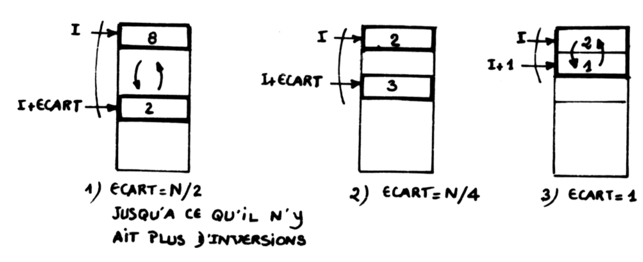
```
10 FOR I=1 TO N-1
20   FOR J=I+1 TO N
30     IF A(J)<A(I) THEN SWAP A(I),A(J)
40   NEXT J
50 NEXT J
```


* COMPARAISON DU I^{ème} AVEC LES AUTRES

Contrairement à la méthode précédente, le tri ne peut être stoppé par le test d'un témoin d'inversion.

SHELL

Avec les méthodes de tri du type Ripple ou Bubble, un grand nombre placé en début de table ne remonte que progressivement en fin de table. Avec Shell, la comparaison s'effectue entre deux éléments séparés par un écart égal, au départ, à la moitié de la taille de la table.




```
100 ECART=N             ' SHELL
105 '
110 ECART=INT(ECART/2):IF ECART<1 THEN STOP
115 '
120 INV=0
130 FOR I=1 TO N-ECART
140   J=I+ECART:IF A(I)>A(J) THEN SWAP A(I),A(J):INV=1
150 NEXT I
160 IF INV=1 THEN 120 ELSE 110
```

SHELL/METZNER

Cette méthode est sans doute celle qui présente le meilleur rapport 'performance/complexité' :

```
100 ECART=N             ' SHELL-METZNER
105 '
110 ECART=INT(ECART/2):IF ECART<1 THEN STOP
120 J=1:K=N-ECART
125 '
130 I=J
140'
160 M=I+ECART
170 IF A(I)<=A(M) THEN 210
180 SWAP A(M),A(I)
190 I=I-ECART:IF I<1 THEN 210 ELSE 160
200 '
210 J=J+1:IF J>K THEN 110 ELSE 130
```

```
10 ' TRIS   28.11.80
20 '                              TRI PAR INSERTION
30 '                              =================
40 '
50 '  On compare le Jeme element (au depart J=2) a tous ceux d'une liste deja
60 '  en ordre(au depart le premier element) et on l'insere a sa bonne position
70 '
80 '
90 '                 -----
100 '                ! 1 !
110 '                ! 2 !
120 '        I ---> ! 3 !
130 '                !10 ! (2)      (3)
140 '                !12 !
150 '                !15 !
160 '                !---!     ----
170 '        J ---> ! 4 ! (1)-> | 4 |
180 '                !---!     ----
190 '                !   !
200 '                !   !
210 '                -----
220 '
230 '
240 INPUT "Nombre? ";N
250 DIM A(N)
260 '-----------------------------------------------------
270 '                              REMPLISSAGE d"UNE TABLE A()
280 FOR I=1 TO N
290   A(I)=INT(RND(1)*1000)
300 NEXT I
310 '----------------------------------------------------------
320 '                              TRI PAR INSERTION
330 FOR J=2 TO N
340   X=A(J)                      ' Sauvegarde de A(J)
350   FOR I=J-1 TO 1 STEP-1       ' Comparaison du Jeme a tous les autres
360      IF X>A(I) THEN 390       ' Les decalages sont finis
370      A(I+1)=A(I)              ' On decale pour inserer
380   NEXT I
390   A(I+1)=X                    ' Insertion
400 NEXT J
410 '---------------------------------------------------------
420 FOR I=1 TO N:PRINT I,A(I):NEXT I
430 '===============================================================
440 '
450 ' Cette methode  est bien adaptee au tri d'une table a laquelle
460 ' on a ajoute des elements en fin de table.
470 '
480 '  Remplacer  FOR J=2 TO N   par  FOR J=N TO N+A
490 '
500 '                          ------
510 '                          !  1 !
520 '                          !  2 !  partie deja en ordre
530 '                          !  3 !
540 '                          !  6 !
550 '                      N   ! 10 !
560 '                          ------
570 '                          !  4 !  ajouts
580 '                    N+A   !  5 !
590 '                          ------
600 ' Le mieux est encore d'inserer chaque nouvel element des qu'il est ajoute a
610 ' la table.
```

```
10 ' TRIPI LE 11/7/80
20 '
30 '                 TRI PAR PERMUTATION D'INDICES
40 '                 -----------------------------
50 '
60 '      On recherche la position  du plus petit de N elements pour l'amener au
70 '      premier rang de la table.
80 '      On procede de la meme facon sur les N-1 elements restants pour amener
90 '      le plus petit de ceux ci en seconde position.
160 '
170 '                 -------
180 '                 !  1  !
190 '          I -->  !  4  !
200 '                 !  6  !
210 '          J -->  !  3  !
220 '                 !  8  !
230 '
240 '------------------------------------------------------------------------
250 '                                   REMPLISSAGE D'UNE TABLE A()
260 '                                     AVEC DES NOMBRES ALEATOIRES
270 DEFINT I-Z
280 INPUT "TAILLE ? ";TAILLE
290 DIM A(TAILLE)
300 FOR I=1 TO TAILLE:A(I)=RND(1):PRINT A(I):NEXT I
310 '------------------------------------------------------------------------
320 '
330 '                                                   TRI
340 FOR I=1 TO TAILLE-1
350   PPETIT=I                        ' Position du plus petit
360   FOR J=I+1 TO TAILLE             ' Recherche du plus petit de I a TAILLE
370     IF A(J)<A(PPETIT) THEN PPETIT=J
380   NEXT J
390   SWAP A(PPETIT),A(I)             ' Rangement du plus petit en I
400 NEXT I
410 '------------------------------------------------------------------------
420 '                                                 EDITION
430 PRINT
440 FOR I=1 TO TAILLE:PRINT I,A(I):NEXT I
```

METHODE DE TRI RAPIDE (QUICKSORT)

L'idée est la suivante :

On répartit la suite de nombres à trier de telle sorte que tous les éléments inférieurs à un élément médian (qui reste à déterminer) soient à gauche de celui-ci et que tous ceux qui lui sont supérieurs soient à sa droite.

```
[70 61 16 48 29 18 59 20 74 36 3 70 0 3 22 39 59 30 58 10]   AVANT
                            ↑
              element median de reference
                            ↓
[3 30 16 22 29 18 3 20 0 10][36][70 74 59 48 39 59 61 58 70] APRES

    elements<=36                      elements>36
```

En procédant ainsi successivement sur les sous-ensembles générés, on obtient une suite d'ensembles ordonnés.

On observe qu'au moment où la taille des sous-ensembles devient inférieure à 10, il est plus rapide d'achever le tri par une méthode classique (Ripple ou Insertion) que de poursuivre la partition.

```
  E1<E2     E2<E3     E3<E4     E4<E5   E5<E6
!--------!-----------!-------!----------!------!
```

Choix de l'élément médian de référence :

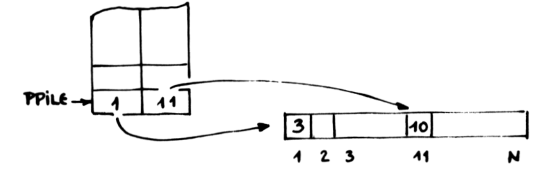
On choisit un élément médian parmi trois éléments : ceux de gauche, du milieu et de droite, de façon à répartir avec un élément ni trop petit, ni trop grand.

A l'issu de ce choix, l'élément médian de référence est rangé en première position de l'ensemble traité.

Pile des adresses des partitions à traiter :

Nous ne pouvons partitionner qu'un ensemble à la fois; les adresses des partitions qui restent à traiter sont stockées dans une pile.

Celle-ci doit avoir une taille d'autant plus grande que le nombre d'éléments à trier est important et que la limite inférieure de partition est faible.



* PILE DES ADRESSES DES PARTITIONS A TRAITER

Exemple avec LINF=4 :

```
                       70 61 16 48 29 18 59 20 74 [36] 3 70 0 3 22 39 59 30 58 10  DEPART

r=36 i=1 j=21          [36] 61 16 48 29 18 59 20 74 10 3 70 0 3 22 39 59 30 58 70  SUITE TRAITEE
                       [3 30 16 22 29 18 3 20 0 10][36][70 74 59 48 39 59 61 58 70] RESULTAT

r=70 i=12 j=21         3 30 16 22 29 18 3 20 0 10 36 [70] 74 59 48 39 59 61 58 70  SUITE TRAITEE
                       3 30 16 22 29 18 3 20 0 10 36 [58 70 59 48 39 59 61][70][74] RESULTAT

r=58 i=12 j=19         3 30 16 22 29 18 3 20 0 10 36 [58] 70 59 48 39 59 61 70 74  SUITE TRAITEE
                       3 30 16 22 29 18 3 20 0 [10] 36 [48 39][58][59 70 59 61] 70 74 RESULTAT

r=10 i=1 j=11          [10] 30 16 22 3 18 3 20 0 29 36 48 39 58 59 70 59 61 70 74  SUITE TRAITEE
                       [3 0 3][10][22 18 16 20 30 29] 36 48 39 58 59 70 59 61 70 74  RESULTAT

r=22 i=5 j=11          3 0 3 10 [22] 18 16 20 30 29 36 48 39 58 59 70 59 61 70 74
                       3 0 3 10 [20 18 16][22][30 29] 36 48 39 58 59 70 59 61 70 74
```

TRI FINAL PAR INSERTION

```
10 ' QUICKSORT (d'apres HOARE adapte par B.BESSE)
20 '
30 '    7 4 15 8 ...... 12 ....3 6 16 2
40 '    !       !         !     !       !
50 '   GAUC     I       MILIEU  J      DROI
60 '
70 DEFINT A-Z:T=256:DIM V(T)
90 '--------------------- Sequence d'essai
100 FOR I=1 TO T:V(I)=RND(1)*1000:NEXT I    ' Remplissage de V() avec nb aleatoires
105 PRINT "TOP"
120 '------------------
130 DIM PILE(1,14)         ' Pile pour stockage des adr des partitions a traiter
150 PPILE=0                ' Pointeur pile
160 LINF=10                ' Limite inférieure de partition
170 '
180 IF T=1 THEN 660
190 IF T<=LINF THEN 540
200 '
210 GAUC=1:DROI=T
220 '------------  Choix de l'element de reference pour la partition
230 MILIEU=INT((GAUC+DROI)/2)
250 IF V(GAUC)<V(MILIEU) THEN SWAP V(GAUC),V(MILIEU)
260 IF V(DROI)<V(GAUC) THEN SWAP V(DROI),V(GAUC):IF V(GAUC)<V(MILIEU) THEN SWAP V(G
AUC),V(MILIEU)
280 '---------------------------------- Partition
290 REF=V(GAUC):I=GAUC:J=DROI+1
310 '
320 I=I+1:IF V(I)<REF THEN 320
330 '
340 J=J-1:IF V(J)>REF THEN 340
350 '
360 IF I<J THEN SWAP V(I),V(J):GOTO 320
370 '
380 SWAP V(GAUC),V(J)
400 '---------------- Chargement de la pile
410 IF (J-GAUC)>(DROI-J) THEN 470
420 '------------------------------------ Partition gauche la + petite
430 IF (J-GAUC)>LINF THEN PPILE=PPILE+1:PILE(0,PPILE)=J+1:PILE(1,PPILE)=DROI:DROI=J
-1:GOTO 230
440 IF (DROI-J)>LINF THEN GAUC=J+1:GOTO 230
450 GOTO 500
460 '----------------------------------- Partition Droite la + petite
470 IF (DROI-J)>LINF THEN PPILE=PPILE+1:PILE(0,PPILE)=GAUC:PILE(1,PPILE)=J-1:GAUC=J
+1:GOTO 230
480 IF (J-GAUC)>LINF THEN DROI=J-1:GOTO 230
490 '----------------------------------- On retire de la pile
500 IF PPILE=0 THEN 540
520 GAUC=PILE(0,PPILE):DROI=PILE(1,PPILE):PPILE=PPILE-1:GOTO 230
530 '----------------------------------- Tri final  par insertion
540 IF V(1)>V(2) THEN SWAP V(1),V(2)
550 IF T=2 THEN 660
560 FOR N=3 TO T
570   IF V(N)>=V(N-1) THEN 630
580   SWAP V(N),V(N-1)
590     FOR K=N-2 TO 1 STEP-1
600       IF V(K)<=V(K+1) THEN 630
610       SWAP V(K),V(K+1)
620     NEXT K
630 NEXT N
640 '-------------------- Edition des resultats
660 FOR K=1 TO T:PRINT V(K):NEXT K
670 '
680 ' 32 s pour 256 nombres aleatoires.20s pour liste triee ou inverse
```

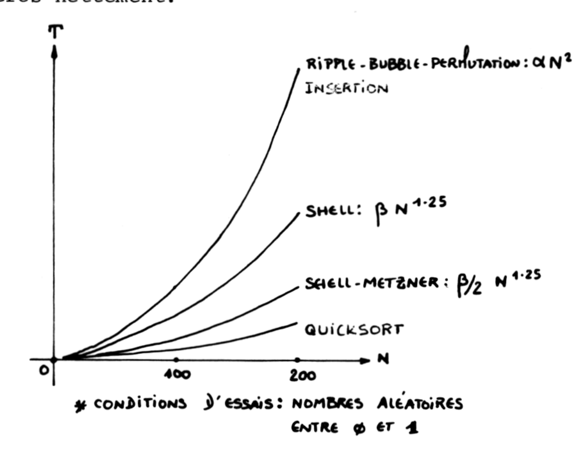
```
10 'QUICKSORT (element reference choisi a gauche et sans tri final)
30 DEFINT A-Z:T=256:DIM V(T+1)
50 '---------------------- Sequence d'essai
60 FOR I=1 TO T:V(I)=RND(1)*1000:PRINT V(I);:NEXT I
65 V(0)=-10000:V(T+1)=10000          ' Bornes
80 '------------------
90 DIM PILE(1,14)           ' Pile pour stockage des adr des partitions a traiter
100 PPILE=0                 ' Pointeur pile
120 GAUC=1:DROI=T
130 '-------------------
140 IF DROI<=GAUC THEN 320
150 '---------------------------------- Partition
160 REF=V(GAUC):I=GAUC:J=DROI+1
170 '
180 I=I+1:IF V(I)<REF THEN 180
190 '
200 J=J-1:IF V(J)>REF THEN 200
210 '
220 IF I<J THEN SWAP V(I),V(J):GOTO 180
240 SWAP V(GAUC),V(J)
250 '----------------- Chargement de la pile
260 IF (J-GAUC)>(DROI-J) THEN 300
270 '------------------------------------ Partition gauche la + petite
280 PPILE=PPILE+1:PILE(0,PPILE)=J+1:PILE(1,PPILE)=DROI :DROI=J-1:GOTO 140
290 '----------------------------------- Partition Droite la + petite
300  PPILE=PPILE+1:PILE(0,PPILE)=GAUC:PILE(1,PPILE)=J-1  : GAUC=J+1:GOTO 140
310 '----------------------------------- On retire de la pile
320 IF PPILE=0 THEN 350
330 GAUC=PILE(0,PPILE):DROI=PILE(1,PPILE):PPILE=PPILE-1:GOTO 140
340 '--------------------- Edition des resultats
350 FOR K=1 TO T:PRINT V(K);:NEXT K
360 '
370 ' 40 sec pour 256 nombres aleatoires.Diverge pour liste triee ou inverse:360s
380 '=====================================================================
```

```
10 ' QUICKSORT (el ref choisi au milieu et sans choix de la partition a traiter)
30 DEFINT A-Z:T=256:DIM V(T+1)
40 '---------------------- Sequence d'essai
50 FOR I=1 TO T:V(I)=RND(1)*1000::NEXT I
80 '------------------
90 DIM PILE(1,20)           ' Pile pour stockage des adr des partitions a traiter
100 PPILE=0                 ' Pointeur pile
120 GAUC=1:DROI=T
130 '-------------------
140 IF DROI<=GAUC THEN 300
150 '---------------------------------- Partition
160 REF=V(INT((GAU+DROI)/2)):I=GAUC:J=DROI
180 '
190 IF V(I)<REF THEN I=I+1:GOTO 190
200 '
210 IF V(J)>REF THEN J=J-1:GOTO 210
220 '
230 IF I>J THEN 270
240 SWAP V(I),V(J):I=I+1:J=J-1:IF I<=J THEN 190
250 '
260 '----------------- Chargement de la pile
270 IF I<DROI THEN PPILE=PPILE+1:PILE(0,PPILE)=I:PILE(1,PPILE)=DROI
280 DROI=J:GOTO 140
290 '----------------------------------- On retire de la pile
300 IF PPILE=0 THEN 330
310 GAUC=PILE(0,PPILE):DROI=PILE(1,PPILE):PPILE=PPILE-1:GOTO 140
320 '--------------------- Edition des resultats
340 '
350 ' 45s pour 256 nombres aleatoires.26s pour liste triee et inverse
```

COMPARAISONS DES TRIS

Lorsque le nombre d'éléments à trier est inférieur à 50, toutes les méthodes de tri sont pratiquement équivalentes.; On choisira donc dans ce cas la plus simple à écrire.

Au-delà de 50, Shell, Shell-Metzner et Quicksort se démarquent très nettement.



TRI DE CHAINES DE CARACTERES

On sait qu'en Basic Microsoft, il est réservé un espace particulier pour les chaînes de caractères (par CLEAR xxx).

On observe qu'avec l'instruction SWAP, les chaînes de caractères ne sont pas déplacées alors qu'au contraire, l'échange de deux chaînes par :

```
X$=A$(I):A$(I)=A$(I+1):A$(I+1)=X$
```

provoque une réallocation des chaînes.

Par conséquent, il y a périodiquement réorganisation de l'espace chaîne et donc un temps d'attente non négligeable qui vient s'ajouter au temps de tri lui-même. Plus le taux d'occupation pour les chaînes est faible, moins les réorganisations sont fréquentes.

L'échange de deux chaînes XØ et YØ, sans SWAP, peut être fait par l'échange des descripteurs de chaînes. Ces descripteurs comportent trois octets qui représentent la longueur et l'adresse des chaînes.

```
100 AX=VARPTR(X$):AY=VARPTR(Y$)
110 FOR I=0 TO 2
120  X=PEEK(AX+I):POKE AX+I,PEEK(AY+I):POKE AY+I,X
130 NEXT I
```

En 'développant' la boucle 'FOR', l'échange des descripteurs serait plus rapide.

```
10 'TCHAI       TRI DE CHAINES DE CARACTERES SANS 'SWAP'
20 '
30 '   On observe que les chaines sont deplacees en memoire centrale
40 '
50 '
60 CLEAR(100)
70 FOR I=1 TO 5
80  A$(I)=STRING$("A",6-I)          ' A$(1)='AAAAA'
90  LPRINT A$(I)
100 NEXT I
110 LPRINT
120 '-------------------------------------------------- TRI
130 FOR I=1 TO 5-1
140    FOR J=I+1 TO 5
150     IF A$(I)>A$(J) THEN  X$=A$(I):A$(I)=A$(J):A$(J)=X$
160   NEXT J
170 '
180   LPRINT "  adr descr  L  adr chaine":LPRINT
190   FOR K=1 TO 5                ' Visualisation des descripteurs de chaines
200     X=VARPTR(A$(K)):LPRINT K;TAB(5);X;PEEK(X);PEEK(X+2);PEEK(X+1);A$(K)
210   NEXT K
220   LPRINT
230 NEXT I
```

30917 LONGUEUR
ADRESSE → AAAAA A$(1)
30920 LONGUEUR
ADRESSE → AAAA A$(2)

DESCRIPTEURS DE CHAINES POUR A$()

```
AAAAA
AAAA
AAA
AA
A

  adr descr  L  adr chaine

 1    30917  1  191  204 A
 2    30920  5  191  226 AAAAA
 3    30923  4  191  215 AAAA
 4    30926  3  191  207 AAA
 5    30929  2  191  202 AA

  adr descr  L  adr chaine

 1    30917  1  191  204 A
 2    30920  2  191  172 AA
 3    30923  5  191  188 AAAAA
 4    30926  4  191  177 AAAA
 5    30929  3  191  169 AAA

  adr descr  L  adr chaine

 1    30917  1  191  254 A
 2    30920  2  191  252 AA
 3    30923  3  191  228 AAA
 4    30926  5  191  235 AAAAA
 5    30929  4  191  224 AAAA

  adr descr  L  adr chaine

 1    30917  1  191  254 A
 2    30920  2  191  252 AA
 3    30923  3  191  228 AAA
 4    30926  4  191  215 AAAA
 5    30929  5  191  210 AAAAA
```

* VARPTR (A$(1)) FOURNIT L'ADRESSE DU DESCRIPTEUR DE A$(1)

TRIS MULTICRITERES

On veut obtenir, à partir d'un fichier client, une liste triée de ceux-ci pour chaque département.

Une méthode simple consiste à effectuer un tri en prenant comme clé la concaténation du département et du nom.

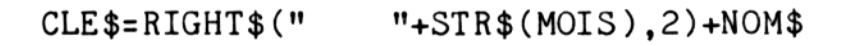
CLE:	78	DUPONT

Si des zones numériques qui n'ont pas un nombre de chiffres constant (mois de naissance par exemple) doivent être concaténées avec des chaînes, il ne suffit pas de les convertir par STR$.
En effet, elles seraient cadrées à gauche.

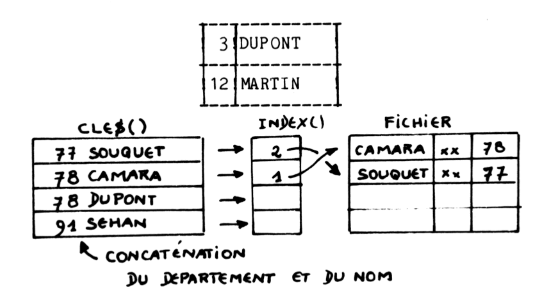
3	DUPONT
12	MARTIN

12MARTIN serait considere comme plus petit que 3DUPONT

On fait donc :



```
CLE$=RIGHT$("    "+STR$(MOIS),2)+NOM$
```



```
10 ' TRIMU 16.11.80        TRI MULTICRITERES
20 '                       -----------------
30 '
140 '
160 '  NOM$    : Nom defini dans fichier 'CLIEN'        15 c
170 '  DEPART$ : Departement defini dans fichier 'CLIEN' 2 c
180 '
190 OPEN "R" ,1,"CLIEN"              ' Ouverture du fichier 'CLIEN'
200 '
210 FIELD #1,15 AS NOM$,12 AS PREN$,20 AS TEL$,2 AS DEPART$
220 '------------------------------------------------------------------
230 '                              CONSTITUTION DES TABLES CLE$() ET INDEX()
240 NCLES=0                                      ' NCLES: Nombre de cles
250 NB=LOF(1):DIM CLE$(NB),INDEX(NB)             ' Tables des CLES et d'INDEX
260 '
270 FOR I=1 TO LOF(1)                            ' Lecture de tout le fichier
280   GET #1,I:IF ASC(NOM$)=0 GOTO 330     ' Lecture de l'enregistrement I
310   NCLES=NCLES+1:CLE$(NCLES)=DEPART$+NOM$:INDEX(NCLES)=I
320   PRINT CLE$(NCLES)
330 NEXT I
340 '--------------------------------------------------------------------
350 TAILLE=NCLES                  ' TRI des tables CLE$() et INDEX()
360 FOR I=1 TO TAILLE-1           '   (par permutation d'indices)
370   PPETIT=I
380   FOR J=I+1 TO TAILLE          ' Recherche du plus petit de I a TAILLE
390     IF CLE$(J)<CLE$(PPETIT) THEN PPETIT=J
400   NEXT J
410 NEXT I
430 '---------------------------------------------------------------------
440 LPRINT "Liste des NOMS tries par DEPARTEMENT"
450 L=LEN(DEPART$)
460 FOR I=1 TO NCLES
470   IF LEFT$(CLE$(I),L)<>LEFT$(CLE$(I-1),L) THEN
       LPRINT:LPRINT "Departement:";LEFT$(CLE$(I),L):LPRINT
480   GET #1,INDEX(I)
490   LPRINT TAB(5) NOM$,PRENOM$,TEL$
500 NEXT I
510 '
```

```
Liste des NOMS tries par DEPARTEMENT

Departement:77

    SOUQUET             Cecile       888-99-00

Departement:78

    CAMARA              Leon         666-99-88
    DUPONT              Jean         777-55-44
    PASQUEREAU          Alain        739-33-90

Departement:91

    SEHAN               Francois     665-13-87
```

CHAPITRE 4

GESTION D'ECRAN

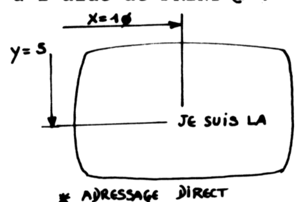
ADRESSAGE DIRECT SUR ECRAN

Les instructions PRINT provoquent une édition continue sur l'écran. Les informations du haut de l'écran disparaissent donc au fur et à mesure de l'exécution de PRINT.

En revanche, l'adressage direct sur écran n'affecte celui-ci qu'aux endroits prévus.

TRS-80 :

Sur TRS-80, l'adressage direct se fait à l'aide de PRINT @ :



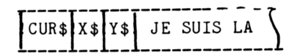

```
10 X=10       ' 10 eme colonne
20 Y=5        ' 5 eme ligne
30'
40 print @Y*64+X,"JE SUIS LA"
```

MICROSOFT 5. :

Il n'existe pas pour le langage Basic Microsoft 5. de fonction d'adressage direct explicite.

C'est par l'envoi d'un caractère de contrôle immédiatement suivi des coordonnées X et Y qu'est assuré l'adressage direct curseur.

```
100 CUR=XX                    ' Valeur specifique a chaque type d'ecran
110 '
120 X=10:Y=5                  ' Abscisse:ordonnee
130 PRINT CHR$(CUR)+CHR$(X)+CHR$(Y);"JE SUIS LA"
```



Les trois caractères avant le texte sont 'interprétés' par l'écran et ne sont bien sûr pas imprimés.

Il est souvent plus pratique de définir une fonction d'adressage curseur qu'il suffit ensuite d'appeler à chaque fois qu'un adressage direct est nécessaire :

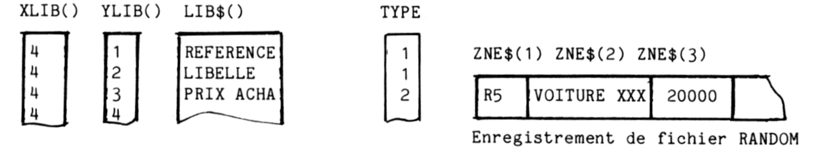
```
50 DEFFNCUR$(X,Y)=CHR$(CUR)+CHR$(X)+CHR$(Y)
60'
100 PRINT FNCUR$(10,5);"JE SUIS LA"
```

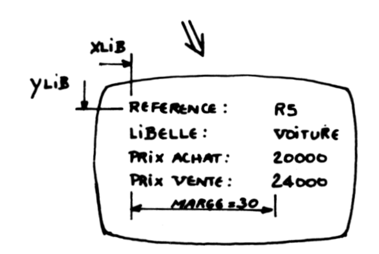
AFFICHAGE D'UN ENREGISTREMENT A L'ECRAN

Soit à afficher sur l'écran différentes zones d'un enregistrement de fichier Random, nous définissons dans différentes tables les paramètres suivants :

- les libellés des zones : table LIB$()
- les types des zones du fichier Random :
  1 --> Chaînes de caractères : table TYPE()
  2 --> Entiers
  3 --> Simple précision
- les coordonnées d'affichage des libellés : tables XLIB() et YLIB()

Nous choisissons d'afficher la zone elle-même avec une marge de 30 par rapport au libellé plutôt que de définir des coordonnées supplémentaires pour celle-ci.






```
10 GOSUB 100:STOP
20 '------------------
100 FOR P=1 TO N
110    PRINT FNCUR$(XLIB(P),YLIB(P));:PRINT LIB$(P);  ' Affichage libelles zones
120    PRINT FNCUR$(XLIB(P)+30,YLIB(P));
130    ON TYPE(P) GOSUB 1000,1010,1020,1030           ' Affichage contenus zones
200 NEXT P
210 RETURN
220 '------------------
210 '
1000 PRINT ZNE$(P):RETURN        ' Chaines
1010 PRINT CVI(ZNE$(P)):RETURN   ' Entiers
1020 PRINT CVS(ZNE$(P)):RETURN   ' Simple precision
```

VIDEO INVERSE

La vidéo inverse qui permet de mieux distinguer certaines zones existe maintenant sur la plupart des écrans. Elle se programme ainsi :

```
100 PRINT CHR$(VIDEO);"REFERENCE:";CHR$(FINVIDEO);REF$
```

SAISIE CARACTERE PAR CARACTERE

La saisie par INPUT ne permet pas de contrôler par programme un caractère dès sa frappe au clavier; il faut attendre que l'opérateur ait frappé un 'retour chariot' pour analyser la ligne frappée et détecter une éventuelle erreur de frappe.

En revanche, INKEY$ sur TRS-80 et INPUT$(1) en MICROSOFT 5. fournissent au programme un caractère juste après sa frappe.

Mais dès lors qu'une saisie est faite caractère par caractère, il appartient au programmeur de gérer tous les caractères frappés, y compris le caractère 'curseur gauche' et 'retour chariot', transparents avec l'instruction INPUT classique.

En outre, les caractères frappés ne sont pas imprimés par Basic, c'est le programmeur qui doit les imprimer (s'ils sont valides).

TRS80:

```
100 GOSUB 1000:STOP                  ' Resultat dans LIGNE$
110 '---------------------------

1000 LIGNE$=""
1005 '
1010 C$=INKEY$:IF C$="" THEN 1010          ' Attente d'un caractere
1020 '
1030 C=ASC(C$):L=LEN(LIGNE$)
1040 IF C=8 THEN IF L>0 THEN LIGNE$=LEFT$(LIGNE$,L-1) ELSE 1010 ' Curseur gauche?
1050 IF C=13 THEN RETURN                                         ' Retour chariot?
1060 LIGNE$=LIGNE$+C$
1070 PRINT C$;
1080 GOTO 1010
```

Remarque : avec INKEY$, le curseur écran n'apparaît plus.

MICROSOFT 5.:

La ligne 1010 devient :

```
1010 C$=INPUT$(1)
```

Attention ! Le Basic Microsoft 5. envoie un retour chariot après l'impression de 80 caractères si aucun retour chariot n'a été programmé. Par conséquent, lors d'une saisie d'écran caractère par caractère, il faut, soit envoyer des retours chariots à l'écran périodiquement, soit programmer une longueur de ligne 'infinie' par l'instruction WIDTH 255.

GENERATEUR DE SAISIE D'ECRAN

Nous vous avions présenté une saisie d'écran où les différents paramètres de saisie (coordonnées des zones à saisir, longueur, type,..) étaient définis dans des tables. Nous vous proposons de définir ces paramètres de façon plus visuelle en 'dessinant' l'écran, par l'intermédiaire de DATA.

Un programme analyse ces DATA et documente des tables XLIB(), YLIB(), LGEUR(), TYPE(), MG(). L'analyse des DATA se fait ligne par ligne à l'aide de la fonction INSTR qui permet de retrouver la position d'un caractère dans une chaîne :

- la première DATA de chaque ligne représente le numéro de ligne à l'écran. Son interprétation est immédiate.
- pour repérer le début et la fin de chaque libellé de zone, nous avons choisi les caractères # et@.

```
#Reference@
```

- la longueur et le type de zone sont définis ainsi :

```
$CCCCCC*
```

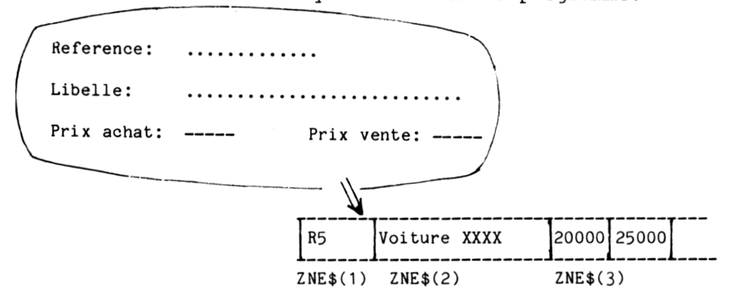
Sur cet exemple, la zone est du type chaîne et doit avoir une longueur maxi de 6.

- la reconnaissance du type de zone se fait par :

```
X=INSTR("CIS",X$) qui fournit X=1,2,3 selon que X$ est C,I,S

(C=chaines I=integer  S=simple precision)
```

Nous générons ensuite le FIELD# du fichier Random où les résultats de la saisie seront rangés. Notons que l'interprétation des DATA est faite à chaque exécution du programme.



Enregistrement de fichier RANDOM

Détails sur la saisie : le retour sur une zone arrière, en cours de saisie, se fait par la touche de code 26 (à adapter au clavier).

1/ Nous affichons une grille de saisie en indiquant par des '.' ou des '-', suivant le type de zone, le nombre de caractères maximum à saisir. Eventuellement, l'ancien contenu de l'enregistrement peut être visualisé (il ne l'est pas sur l'exemple).

2/ Nous saisissons ensuite les informations caractère par caractère en respectant bien entendu la grille précédemment affichée.

Lorsque, en cours de saisie, l'opérateur appuie sur la touche 'curseur gauche', nous remplaçons le dernier caractère frappé par un '.' ou un '-' selon le type de zone.

```
1730 IF ... THEN PRINT CHR$(8);CA$;CHR$(8);    ' CA$="." ou "-"
```

Remarque : En Microsoft 5., l'impression de caractère sans RC (par PRINT C$;) provoque l'envoi automatique d'un RC tous les 120 caractères.

Adaptation TRS-80 : la fonction FNCUR$ est remplacée par PRINT@

C$=INPUT$(1) est remplacé par : 1700 C$=INKEY$:IF C$="" THEN 1700

La gestion du curseur écran peut se faire par :

```
1370 SET(XLIB(P)+MG(P)-1)*2,YLIB(P)*3+1:GOSUB 1640     ' Allumage curseur ligne

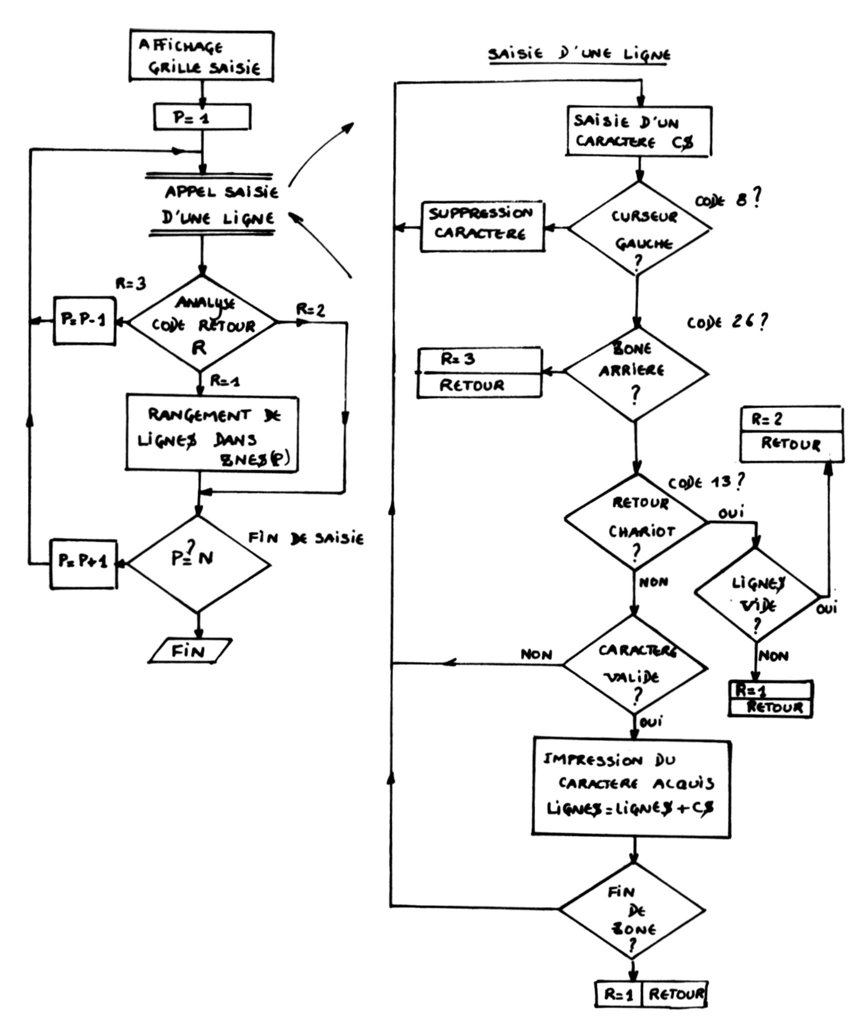
1375 RESET(..,..)                                      ' Extinction curseur ligne
```

Sur TRS-80, l'impression de CHR$(8) provoque l'effacement du dernier caractère affiché à l'écran, on fait donc :

```
1730 IF C=8 .. THEN ..    PRINT CHR$(8);CA$;:PRINT @XY+L-1,"";

1660 XY=YLIB(P)*64+XLIB(P)+MG(P)            ' Calcul de X,Y debut de zone

1660 PRINT @XY,"";                          ' Positionnement debut de zone saisie
```


SAISIE DE PLUSIEURS ZONES
AFFICHAGE GRILLE SAISIE
P= 1
APPEL SAISIE D'UNE LIGNE
R=3
P=P-1
ANALYSE CODE RETOUR R
R=2
R=1
RANGEMENT DE LIGNE$ DANS ZNE$(P)
P=P+1
P=?N
FIN DE SAISIE
FIN
SAISIE D'UNE LIGNE
SAISIE D'UN CARACTERE C$
CODE 8 ?
SUPPRESSION CARACTERE
CURSEUR GAUCHE ?
CODE 26 ?
R=3
RETOUR
ZONE ARRIERE ?
R=2
RETOUR
CODE 13 ?
RETOUR CHARIOT ?
OUI
LIGNE$ VIDE ?
OUI
NON
NON
R=1
RETOUR
NON
CARACTERE VALIDE ?
OUI
IMPRESSION DU CARACTERE ACQUIS
LIGNE$=LIGNE$+C$
FIN DE ZONE ?
R=1
RETOUR

```
82 'SAIZ          GENERATEUR DE SAISIE D'ECRAN
84 '
86 OPEN "R",#1,"HAS"
90 EF$=CHR$(31)                    ' Effacement ecran (a adapter)
100 '----------------------------------- Dessin de l'ecran
120 '  NLIG       p1   p2        p3         p4
130 '     !        !    !         !          !
140 '     v        v    v         v          v
150 DATA 1,"      #Nom::@         $CCCCCCCCCC*"
160 DATA 3,"      #Prenom: :@     $CCCCCCCCCCCCCCCCCCCCCCCCCCC*"
170 DATA 5,"      #Matricule:@    $SSSSS*       #Telephone::@     $CCCCCCCCC*
180 DATA 99
190 '------------------ Constitution de LIB$() XLIB() YLIB() LGEUR() TYPE() MG()
200 P=1
210 '
220 READ NLIG:IF NLIG=99 THEN N=P-1:GOTO 350        ' Lecture du no de ligne ecran
230 READ LIGNE$:DB=1                                ' Lecture d'une ligne ecran
240 '
250 P1=INSTR(DB,LIGNE$,"#"):P2=INSTR(DB,LIGNE$,"@") ' Debut et fin du libelle
260 P3=INSTR(DB,LIGNE$,"$"):P4=INSTR(DB,LIGNE$,"*") ' Debut et fin zone a saisir
270 YLIB(P)=NLIG:XLIB(P)=P1                         ' Coordonnees du libelle
280 LIB$(P)=MID$(LIGNE$,P1+1,P2-P1-1)               ' Libelle
290 LGEUR(P)=P4-P3:MG(P)=P3-P1                      ' Longueur de saisie
300 X$=MID$(LIGNE$,P3+1,1)                          ' Type de zone C,I,S
310 X=INSTR("CIS",X$):TYPE(P)=X           ' 1:chaines 2:Integer 3:Simple precision
320 P=P+1
330 IF INSTR(P4,LIGNE$,"#")=0 THEN 220 ELSE DB=P4+1:GOTO 250
340 '-------------------- Generation du FIELD#
350 D=0
360 FOR I=1 TO N
370   IF TYPE(I)=2 THEN FIELD #1,D AS D$,2 AS ZNE$(I):D=D+2:GOTO 400
380   IF TYPE(I)=3 THEN FIELD#1,D AS D$,4 AS ZNE$(I):D=D+4:GOTO 400
390   FIELD #1,D AS D$,LGUEUR(I) AS ZNE$(I):D=D+LGUEUR(I)
400 NEXT I
410 DEF FNCUR$(X,Y)=CHR$(16)+CHR$(31+Y)+CHR$(31+X)    ' Adressage curseur(a adapter
)
430 PRINT EF$
445 GET #1,1:GOSUB 1320:PUT #1,1:STOP       ' Essai
1300 '====================================== SAISIE DE N ZONES DANS ZNE$()
1310 '
1320 FOR I=1 TO N:TRAV$(I)="":NEXT I        ' Table de travail pour saisie
1330 GOSUB 1930                             ' Appel Edition grille ecran
1340 '
1350 P=1                                    ' P: Zone courante 1,2,3,4,..,N
1360 '
1370 GOSUB 1640                             ' Appel saisie de LIGNE$
1380 ON R GOTO 1420,1450,1400               ' R=1:OK /R=2:ligne vide R=3:On remonte
1390 '
1400 IF P>1 THEN ON TYPE(P) GOSUB 2010,2020,2020:P=P-1:GOTO 1370 ELSE 1370  ' R=3 :
 on remonte
1410 '
1420 ON TYPE(P) GOSUB 1500,1510,1520        ' Appel rangement fichier
1430 TRAV$(P)=LIGNE$
1440 '
1450 PRINT FNCUR$(XLIB(P)+MG(P),YLIB(P));
1460 ON TYPE(P) GOSUB 2010,2020,2020        ' Reaffichage zone
1470 IF P=>N THEN RETURN                    ' Fin de saisie?
1480 P=P+1:GOTO 1370
1490 '
1500 LSET ZNE$(P)=LIGNE$:RETURN             ' Rangement de LIGNE$ dans BUFFER
1510 LSET ZNE$(P)=MKI$(VAL(LIGNE$)):RETURN
1520 LSET ZNE$(P)=MKS$(VAL(LIGNE$)):RETURN
1530 '------------------------------------------- SAISIE D'UNE LIGNE
```

```
1630 '-------------------------------- SAISIE D'UNE LIGNE dans LIGNE$
1640 LIGNE$=""
1650 IF TYPE(P)=1 THEN CA$="." ELSE CA$="-"
1660 PRINT FNCUR$(XLIB(P)+MG(P),YLIB(P));
1670 '
1680 '      Curseur gauche:8 /Retour chariot:13 / Zone arriere:26
1690 '
1700 C$=LCHR$(98)              ' Lecture d'un car au clavier (INPUT$(1) ou INKEY$)
1710 C=ASC(C$):L=LEN(LIGNE$)
1720 '
1730 IF C=8 THEN IF LIGNE$<>"" THEN LIGNE$=LEFT$(LIGNE$,L-1):  PRINT CHR$(8);CA$;CH
R$(8);:GOTO 1700 ELSE 1700
1740 IF C=13 THEN IF LIGNE$<>"" THEN R=1:RETURN ELSE R=2:RETURN
1750 IF C=26 THEN R=3:RETURN                                    ' Retour zone arriere?
1760 ON TYPE(P) GOSUB 1820,1830,1830:ON R GOTO 1770,1700           ' Appel controle
1770 PRINT C$;                                                  ' Affichage car frappe
1780 LIGNE$=LIGNE$+C$
1790 IF L+1>=LGEUR(P) THEN R=1:RETURN               ' Fin de zone?
1800 GOTO 1700
1810 '
1820 IF C=>32 THEN R=1:RETURN ELSE PRINT CHR$(7);:R=2:RETURN              ' Controle
1830 IF C>47 AND C<58 OR C=46 THEN R=1:RETURN ELSE PRINT CHR$(7);:R=2:RETURN
1840 '
1850 '
1910 '-------------------------------------------------------------------
1920 '                                                   AFFICHAGE GRILLE
1930 FOR P=1 TO N          ' N zones
1940   PRINT FNCUR$(XLIB(P),YLIB(P));
1950   PRINT LIB$(P);
1955   ON TYPE(P) GOSUB 2022,2023,2024             ' Anciennes zones
1960   PRINT FNCUR$(XLIB(P)+MG(P),YLIB(P));
1970   ON TYPE(P) GOSUB 2010,2020,2020
1980 NEXT P
1990 RETURN
2000 '
2010 PRINT TRAV$(P);STRING$(".",LGUEUR(P)-LEN(TRAV$(P))):RETURN
2020 PRINT TRAV$(P);STRING$("-",LGUEUR(P)-LEN(TRAV$(P))):RETURN
2021 '
2022 IF ASC(ZNE$(P))<>0 THEN TRAV$(P)=ZNE$(P):RETURN ELSE RETURN
2023 X$=STR$(CVI(ZNE$(P))):TRAV$(P)=LEFT$(X$,LEN(X$)-1):RETURN
2024 X$=STR$(CVS(ZNE$(P))):TRAV$(P)=LEFT$(X$,LEN(X$)-1):RETURN
2030 '---------------------------------------------------------------
2040 '    LIB$()           : Table des libelles de zones
2050 '    XLIB() YLIB()    : Coordonnes des libelles
2060 '    LGEUR()          : Longueur des zones(saisie)
2070 '    MG()             : Marge par rapport a X du libelle
2080 '    TYPE()           : Type de zones  (C,I,S)
2090 '
2100 '----------------------------------------------------------
2110 ' ASSOCIER CE PROGRAMME AVEC 'HASH' pour obtenir un programme de saisie
2120 ' avec acces par cle.
```

CHAPITRE 5

PROGRAMMES

FACTURATION
(cf. Hash-Code et allocation dynamique)

Nous allons montrer sur cet exemple relativement complexe (3 fichiers : clients, produits, factures avec accès par clé) comment on peut, par une programmation modulaire, se passer d'organigramme.

Il s'agit de constituer des factures en se servant d'un fichier client et d'un fichier produit. Ces factures sont ensuite enregistrées dans un fichier, leur édition se faisant ultérieurement.

L'ensemble des fichiers doit être géré dynamiquement, c'est-à-dire, qu'il n'y a pas de mode création explicite pour ajouter un nouveau client ou un nouveau produit. En cours de saisie d'une facture, deux sous-programmes sont chargés de rechercher les clients ainsi que les produits et de les créer dynamiquement s'ils n'existent pas. Ceci évite à l'opérateur de vérifier, avant une facturation, si le client et les produits existent déjà.

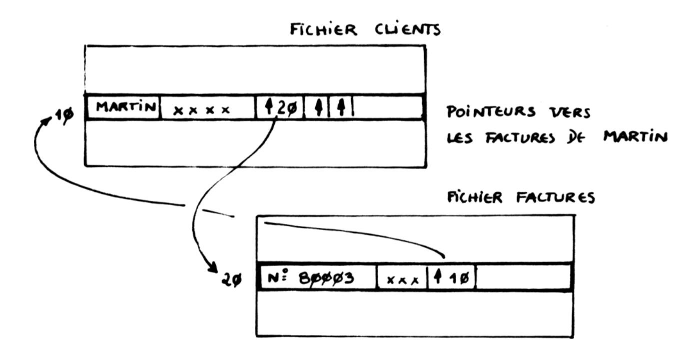
Nous avons prévu un accès par clé pour les clients, les produits et les factures. La méthode utilisée (cf. Hash-Code et allocation dynamique) assure en outre la gestion de l'allocation dynamique, c'est-à-dire, qu'à chaque fois qu'un client, un produit ou une facture sont supprimés, la place qu'ils occupaient est récupérable pour un ajout ultérieur.

Faut-il dans le fichier des factures, placer directement les noms des clients et les noms des produits ou des pointeurs vers les enregistrements vers ceux-ci ?

La seconde méthode a l'avantage d'économiser de la place. En effet, deux octets suffisent pour coder un numéro d'enregistrement. En outre, il n'est pas nécessaire de rechercher les enregistrements des fichiers client et produit via les tables d'index, lors d'une édition de facture par exemple.

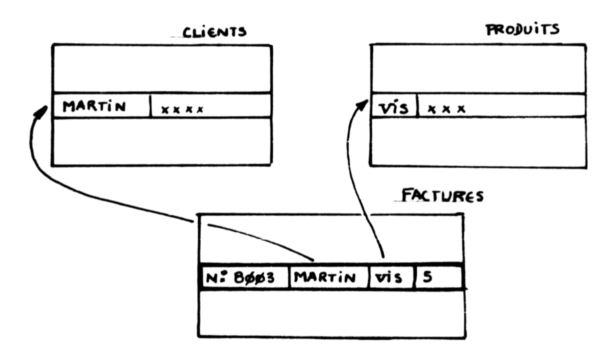
Il faut cependant prendre garde que des clients ou produits vers lesquels des factures pointeraient ne soient supprimés et que d'autres viennent prendre leur place.

Une protection consisterait à gérer des pointeurs 'réciproques'. Avant de supprimer un client, on s'assurerait en temps réel qu'il n'existe pas de pointeur vers au moins une facture, ces pointeurs étant bien entendu supprimés lorsque les factures sont soldées.



Compte tenu du caractère simple que nous avons voulu donner à cet exemple, nous écrivons directement le nom du client et des produits dans la facture.

Les suppressions de clients et de produits ne pourront guère être faites qu'en différé en tenant compte des factures.



Une méthode d'accès par clé plus simple consisterait à rechercher celle-ci par un balayage séquentiel du fichier. Ceci, au détriment du temps d'accès.

```
   INPUT "Produit? ";PX$
   GOSUB PRODUIT
   STOP


PRODUIT:FOR I=1 TO LOF(2)
          GET #2,I
          IF PX$=REF$ THEN Q=1:PRANG=I:RETURN    ' La cle existe
        NEXT I

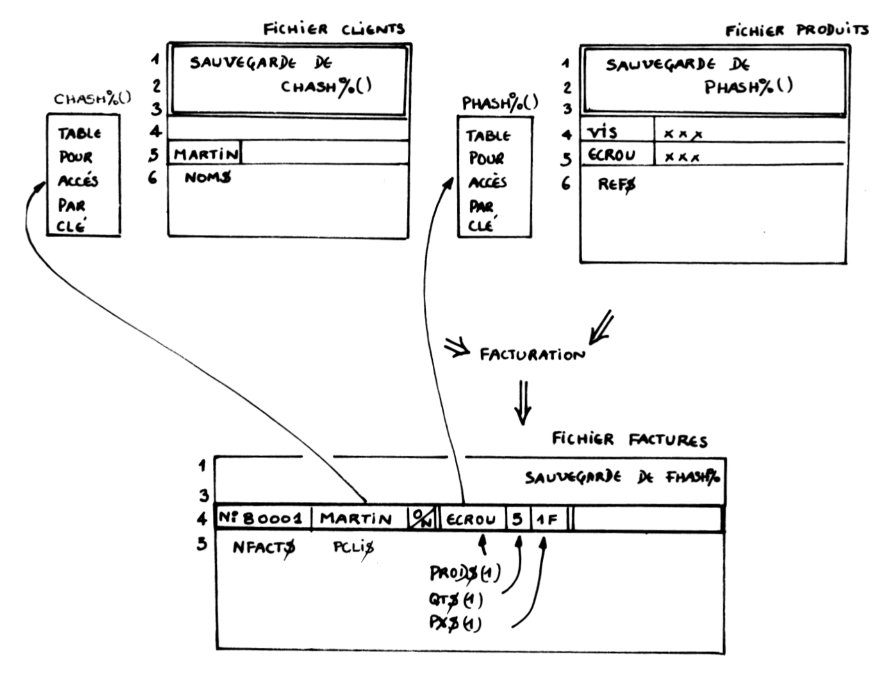
        PRANG=LOF(2)+1:Q=2:RETURN                ' La cle n'existe pas
                                                 ' Allocation en fin de fichier
```

Avec ce système, l'allocation dynamique n'est bien sûr pas assurée, c'est-à-dire, que si un produit est supprimé, la place qu'il occupait ne sera pas récupérable pour un nouvel ajout. C'est volontairement, afin de ne pas surcharger le programme (et ainsi de mieux faire apparaître l'essentiel), que nous n'avons pas traité l'édition de la facture comme il conviendrait de le faire pour un cas réel.

Nous n'avons pas fait figurer le listing des sous-programmes de recherche/création des produits et des factures. Ils correspondent quasiment à ceux de la recherche/création client, aux noms de variables près.

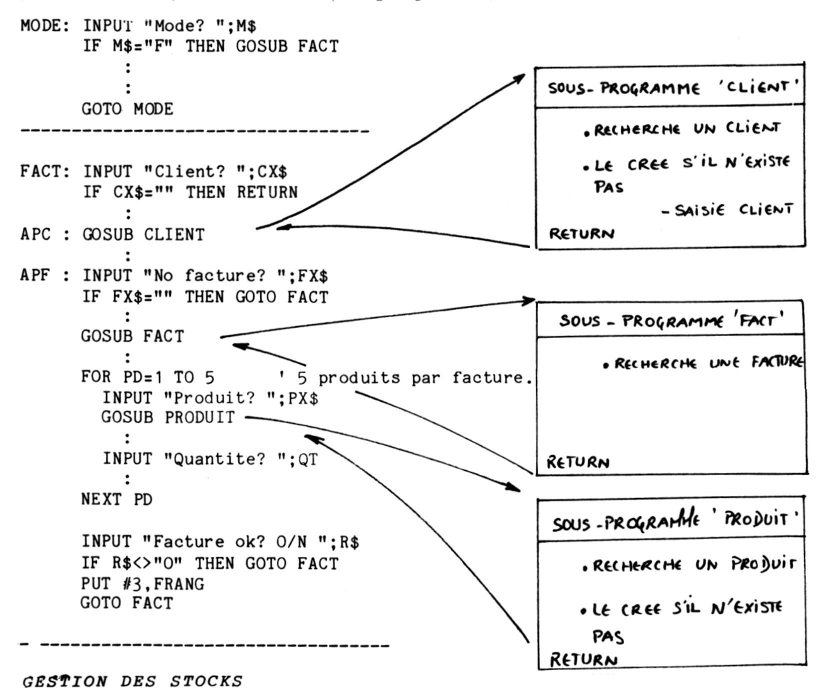
Remarquons que les clients sont créés non seulement dynamiquement (en cours de facturation), mais également directement (mode MC), les sous-programmes de saisie étant bien entendu communs. Il en va de même pour les produits.

FACTURATION



* ETABLISSEMENT D'UNE FACTURE

***FACTURATION** (architecture du programme)*



GESTION DES STOCKS

En différé :

Les stocks peuvent être gérés en *différé* périodiquement en soustrayant les quantités livrées (fournies par les factures) des quantités en stock. Un indicateur (SS$) est alors positionné à 1 dans les factures traitées de façon à se souvenir qu'elles ont déjà été soustraites des stocks.

L'avantage d'une gestion en différé est qu'il suffit, en cas d'incident, (coupure de tension), de reprendre le traitement interrompu en se servant des sauvegardes que l'on a pris soin de faire avant le traitement.

GESTION des STOCKS en DIFFERE

```
5000 FOR F=1 TO LOF(3)                          ' Toutes les factures
5010    GET #3,F
5020    IF SS$="O" THEN GOTO 5100               ' Facture deja traitee
5030    FOR P=1 TO 5                            ' 5 Produits possibles par facture
5040       GOSUB RECHERCHE-PRODUIT              ' Fournit PRANG
5050       LSET STCK$=MKI$(CVI(STCK$)-cvi(QT$(P))) ' MAJ stock
5060       PUT #2,PRANG                         ' PRANG: Adresse de rangement produit
5070    NEXT P
5080    LSET SS$="O"                            ' Facture soustraite du stock.
5090    PUT #3,F
5100 NEXT F
```

En revanche, les stocks ne sont pas connus en *'temps réel'* (Naturellement, une facture ne doit pas être supprimée du fichier avant d'avoir été soustraite du stock).

En temps réel :

Une gestion en temps réel demande quelques précautions, car, en cas d'arrêt intempestif de la machine, il risque de ne plus y avoir cohérence entre les factures et les stocks. Comment donc s'y retrouver dans ce cas ?

VIS	200	150	
	stock initial	stock reel	

En plus de la zone mise à jour en temps réel, une zone contient un stock dit 'initial' périodiquement mis à jour en différé comme ci-dessus ou seulement lorsque les factures soldées sont supprimées du fichier.

Si un incident survient (en temps réel), la zone stock 'temps réel' peut être réactualisée en différé grâce au stock initial et aux factures.

On évitera de procéder à la mise à jour des stocks en cours de constitution de la facture. C'est seulement lorsque la facture sera considérée comme "bonne" que l'on procèdera aux mises à jour des produits.

```
10 ' FAC 12.12.80
20 '
30 '                    FACTURATION SIMPLIFIEE SANS ACCES PAR CLE
32 '
35 '  Les fichiers CLIENT et PRODUIT sont deja constitues
36 '  Ce programme genere simplement des factures qui sont enregistrees
37 '  dans un fichier des factures(en fin de fichier)
40 '
170 OPEN "R",#1,"CLIEN"
180 OPEN "R",#2,"PROD"
190 OPEN "R",#3,"FACT"
200 '------ fichier clients
210 FIELD #1,15 AS NOM$,20 AS RUE$,15 AS VILLE$,5 AS CPOST$
230 '------ fichier produits
240 FIELD #2,12 AS REF$,25 AS LIB$,4 AS PACHA$,4 AS PVENTE$,4 AS QV$,4 AS STK$
260 '------ fichier factures
270 FIELD #3,8 AS NFACT$,1 AS JOUR$,1 AS MOIS$,1 AS AN$,15 AS PCLI$,1 AS SOLD$
280 FOR I=1 TO 5
290   FIELD #3,30 AS D$,(12+2+4)*(I-1) AS D$,12 AS PPROD$(I),2 AS QT$(I),4 AS PRIX$
(I)
300 NEXT I
320 '
360 '----------------------------------------------------------
370 '                                    Menu
380 INPUT "Mode? (F,LF,MC,.. ";M$
390 IF M$="F" THEN GOSUB 760      ' Constitution d'une facture
400 IF M$="LF" THEN GOSUB 1020    ' Edition  d'une facture
410 IF M$="MC" THEN GOSUB 1720    ' Modification/creation client directe
420 GOTO 380
430 '====================================================================
740 '                         CONSTITUTION D'UNE FACTURE
750 '
760 PRINT:INPUT "Client? ";CX$
770 IF CX$="" THEN RETURN
780 GOSUB 1310                       ' Appel recherche client
790 IF R=3 THEN GOTO 760             ' Annulation?
800 PRINT:INPUT "Facture? ";FX$
810 IF LEN(FX$)<5 THEN 760
820 '
830 FRANG=LOF(3)                     ' Rgmt en fin de fichier  LOF(3)+1 sur TRS80
832 GET #3,FRANG:LSET NFACT$=FX$
840 LSET PCLI$=NOM$                  ' Pointeur vers client
850 '--------------
860 '               Saisie produits pour la facture.
870 FOR PD=1 TO 5                    ' 5 produits possibles par facture
880 PRINT:  INPUT "Produit? ";PX$
890   IF LEN(PX$)<3 THEN 960
900   GOSUB 2390                     ' Appel recherche produit
910   LSET PPROD$(PD)=REF$           ' Pointeur vers produit
920   INPUT "Quantite? ";QT
930   LSET QT$(PD)=MKI$(QT)
940   PRINT CVS(PVENTE$);:INPUT "Prix? ";PRIX:IF PRIX<>0 THEN LSET PRIX$(PD)=PVENTE
$ ELSE LSET PRIX$(PD)=PVENTE$ 'Prix standard ou non?
950 NEXT PD
960 PRINT:INPUT "Facture OK? O/N ";R$:IF R$<>"O" THEN RETURN
970 PUT #3,FRANG
980 GOTO 760
990 '====================================================================
```

```
1010 '                              Edition d'une FACTURE (quantite et prix)
1020 INPUT "Facture? ";FX$
1030 IF LEN(FX$)<5 THEN RETURN
1040 FOR FRANG=1 TO LOF(3)
1041   GET #3,FRANG
1042   IF FX$=LEFT$(NFACT$,LEN(FX$)) THEN GOTO 1060
1044 NEXT FRANG
1046 PRINT "Facture n'existe pas":GOTO 1020
1047 '
1050 '
1060 CX$=PCLI$:GOSUB 1310:ON R GOTO 1080,1020   ' Appel recherche client
1070 '
1080 GET #1,CRANG
1090 PRINT:PRINT NOM$
1100 PRINT:PRINT TAB(3);RUE$:PRINT TAB(6);CPOST$;"  ";VILLE$
1110 '
1120 PRINT
1130 FOR PD=1 TO 5                            ' 5 produits par facture
1140   IF ASC(PPROD$(PD))=0 THEN 1200
1150   PX$=PPROD$(PD):GOSUB 2400              ' Appel recherche produit
1160 '
1170   GET #2,PRANG
1180   PRINT TAB(20);REF$,CVI(QT$(PD));
1190   PRINT CVS(PRIX$(PD))
1200 NEXT PD
1210 GOTO 1020
1220 '=========================================================================
1260 '
1270 '
1290 '              RECHERCHE CLIENT
1300 '
1310 FOR CRANG=1 TO LOF(1)
1312   GET #1,CRANG
1314   IF CX$=LEFT$(NOM$,LEN(CX$)) THEN R=1:PRINT NOM$:RETURN
1316 NEXT CRANG
1320 R=3:PRINT "Ce nom n'existe pas"::RETURN
1700 RETURN
1900 '
1910 '
1920 '
2370 '=========================================================================
2380 '
2390 '          RECHERCHE PRODUIT
2395 '
2400 FOR PRANG=1 TO LOF(2)
2410   GET #2,PRANG
2412   IF PX$=LEFT$(REF$,LEN(PX$)) THEN PRINT REF$:R=1:RETURN
2414 NEXT PRANG
2416 PRINT "Produit n'existe pas":R=3:RETURN
```

```
10 ' FACTU 12.12.80
20 '
30 '                    FACTURATION
40 '
50 NCLES=40                  'Nombre de cles par enreg pour sauvegarde des tables CHA
SH%,PHASH%,FHASH%
60 DIM CHASH$(NCLES)         ' Pour FIELD#1  sauvegarde de CHASH%
70 DIM PHASH$(NCLES)         ' Pour FIELD#2  sauvegarde de PHASH%
80 DIM FHASH$(NCLES)
90 '
100 TCLES=NCLES*3           ' Nombre de cles maxi
110 DIM CHASH%(TCLES)       ' Table CHASH% pour l'acces aux clients
120 DIM FHASH%(TCLES)       ' Table FHASH% pour l'acces aux factures
130 DIM PHASH%(TCLES)
140 '
150 ' Sauvegarde de CHASH%,FHASH%,PHASH%  dans les 3 premiers secteurs de chaque fi
chier
160 '
170 OPEN "R",#1,"CLIEN"
180 OPEN "R",#2,"PROD"
190 OPEN "R",#3,"FACT"
200 '------ fichier clients
210 FIELD #1,15 AS NOM$,20 AS RUE$,15 AS VILLE$,5 AS CPOST$
220 FOR I=1 TO NCLES:FIELD #1,2*(I-1) AS D$,2 AS CHASH$(I):NEXT I
225 FIELD #1,(NCLES*2) AS I1$
230 '------ fichier produits
240 FIELD #2,12 AS REF$,25 AS LIB$,4 AS PACHA$,4 AS PVENTE$,4 AS QV$,4 AS STK$
250 FOR I=1 TO NCLES:FIELD #2,2*(I-1) AS D$,2 AS PHASH$(I):NEXT I
255 FIELD #2,(NCLES*2) AS I2$
260 '------ fichier factures
270 FIELD #3,8 AS NFACT$,1 AS JOUR$,1 AS MOIS$,1 AS AN$,15 AS PCLI$,1 AS SOLD$
280 FOR I=1 TO 5
290   FIELD #3,30 AS D$,(12+2+4)*(I-1) AS D$,12 AS PPROD$(I),2 AS QT$(I),4 AS PRIX$
(I)
300 NEXT I
310 FOR I=1 TO NCLES:FIELD #3,2*(I-1) AS D$,2 AS FHASH$(I):NEXT I
315 FIELD #3,(NCLES*2) AS I3$
320 '
330 GOSUB 1650      ' Appel lecture table CHASH% en memoire centrale
340 GOSUB 2290      ' Appel lecture table FHASH% en memoire centrale
350 GOSUB 2810      ' Appel lecture PHASH%
360 '-----------------------------------------------------------
370 '                                      Menu
380 INPUT "Mode? (F,LF,MC,.. ";M$
390 IF M$="F" THEN GOSUB 760      ' Constitution d'une facture
400 IF M$="LF" THEN GOSUB 1020    ' Edition  d'une facture
410 IF M$="MC" THEN GOSUB 1720    ' Modification/creation client directe
420 GOTO 380
430 '=====================================================================
```

```
720 '
730 '=========================================================================
740 '                    CONSTITUTION D'UNE FACTURE
750 '
760 PRINT:INPUT "Client? ";CX$
770 IF CX$="" THEN RETURN
780 GOSUB 1310                      ' Appel recherche client
790 IF R=3 THEN GOTO 760            ' Annulation?
800 PRINT:INPUT "Facture? ";FX$
810 IF LEN(FX$)<5 THEN 760
820 '
830 GOSUB 1950                      ' Appel recherche facture(FRANG en retour)
840 LSET PCLI$=NOM$                 ' Pointeur vers client
850 '-------------
860 '             Saisie produits pour la facture.
870 FOR PD=1 TO 5                   ' 5 produits possibles par facture
880 PRINT:  INPUT "Produit? ";PX$
890   IF LEN(PX$)<3 THEN 960
900   GOSUB 2390                    ' Appel recherche produit
910   LSET PPROD$(PD)=REF$          ' Pointeur vers produit
920   INPUT "Quantite? ";QT
930   LSET QT$(PD)=MKI$(QT)
940   PRINT CVS(PVENTE$);:INPUT "Prix? ";PRIX:IF PRIX<>0 THEN LSET PRIX$(PD)=PVENTE
$ ELSE LSET PRIX$(PD)=PVENTE$ 'Prix standard ou non?
950 NEXT PD
960 PRINT:INPUT "Facture OK? O/N ";R$:IF R$<>"O" THEN RETURN
970 PUT #3,FRANG
980 GOTO 760
990 '=========================================================================
1000 '
1010 '                             Edition d'une FACTURE (quantite et prix)
1020 INPUT "Facture? ";FX$
1030 IF LEN(FX$)<5 THEN RETURN
1040 GOSUB 2050:ON R GOTO 1060,1020           ' Appel recherche facture
1050 '
1060 CX$=PCLI$:GOSUB 1440:ON R GOTO 1080,1020   ' Appel recherche client
1070 '
1080 GET #1,CRANG
1090 PRINT:PRINT NOM$
1100 PRINT:PRINT TAB(3);RUE$:PRINT TAB(6);CPOST$;"  ";VILLE$
1110 '
1120 PRINT
1130 FOR PD=1 TO 5                           ' 5 produits par facture
1140   IF ASC(PPROD$(PD))=0 THEN 1200
1150   PX$=PPROD$(PD):GOSUB 2570             ' Appel recherche produit
1160 '
1170   GET #2,PRANG
1180   PRINT TAB(20);REF$,CVI(QT$(PD));
1190   PRINT CVS(PRIX$(PD))
1200 NEXT PD
1210 GOTO 1020
1220 '=========================================================================
```

```
1290 '                 RECHERCHE CLIENT   (le cree s'il n'existe pas)
1300 '
1310 GOSUB 1440:ON R GOTO 1320,1340            ' Appel recherche cle
1320 PRINT NOM$:RETURN                         ' La cle existe
1330 '
1340 INPUT "NOUVEAU NOM OK? ";R$:IF R$<>"O" THEN R=3:RETURN
1350 LSET I1$=STRING$(CHR$(0),NCLES*2)         ' Initialisation buffer avec 0
1360 LSET NOM$=CX$:GOSUB 1850                  ' Appel saisie
1370 PUT #1,CRANG:CHASH%(CRANG)=CCLE:GOSUB 1590 ' Appel sauvegarde CHASH%
1380 GET #1,CRANG:RETURN                       ' Rappel client(indispensable)
1390 '-------------------- Recherche dela cle
1400 '
1410 ' Entree:CX$    Retour:   R=1  :  La cle existe /R=2 :N'existe pas
1420 '                         CRANG : Adresse de rangement
1430 '                         NOM$ : Nom du client
1440 FOR I=1 TO 3:X(I)=ASC(MID$(CX$,I,1))-64:NEXT I
1450 CCLE=X(1)*26*26+X(2)*26+X(3)              ' Calcul d'une cle numerique
1460 CLIB=0                                    ' Position libre dans CHASH%
1470 FOR I%=3+1 TO TCLES
1480   IF CHASH%(I%)=0 THEN  GOTO 1550
1490   IF CHASH%(I%)<>CCLE THEN 1520
1500   GET #1,I%:IF CX$=LEFT$(NOM$,LEN(CX$)) THEN CRANG=I%:R=1:RETURN
1510 '
1520   IF CLIB=0 THEN IF CHASH%(I%)=-32000 THEN CLIB=I%     ' -32000:libre
1530 NEXT I%
1540 PRINT "C'est plein":STOP
1550 IF CLIB=0 THEN CLIB=I%
1560 R=2:CRANG=CLIB:RETURN
1570 '---------------------- Sauvegarde table CHASH% (morceau modifie)
1590 DB=INT((CRANG-1)/NCLES)    ' DB: No du bloc de CHASH% a sauvegader(0,1,2)
1600 NB=DB*NCLES:GET #1,DB+1
1610 FOR J=1 TO NCLES:NB=NB+1:LSET CHASH$(J)=MKI$(CHASH%(NB)):NEXT J
1620 PUT #1,DB+1
1630 RETURN
1640 '---------------------------------- Lecture table CHASH%
1650 NB=0:IF LOF(1)=0  THEN LSET I1$=STRING$(CHR$(0),NCLES*2):FOR I=1 TO 3:PUT #1,I
:NEXT I:LSET CHASH$(1)=MKI$(32000):PUT #1,1  ' Initialisation index avec 0 ASCII
1660 FOR I=1 TO 3
1680    GET #1,I:FOR J=1 TO NCLES:NB=NB+1:CHASH%(NB)=CVI(CHASH$(J)):NEXT J
1690 NEXT I
1700 RETURN
1710 '=====================================================================
1720 INPUT "Nom? ";CX$:IF CX$="" THEN RETURN        ' CREATION/MODIFICATION(MC)
1730 GOSUB 1440:ON R GOTO 1750,1760                 ' Appel recherche cle
1740 '                                              ' Le client existe deja
1750 GOSUB 1850:PUT #1,CRANG:GOTO 1720              ' Appel saisie/modification
1760 '------------- Le client n'existe pas
1770 INPUT "Nouveau client? (O/N) ";R$:IF R$<>"O" THEN 1720
1780 LSET I1$=STRING$(CHR$(0),NCLES*2)   ' Initialisation buffer avec 0 ASCII
1790 LSET NOM$=CX$:GOSUB 1850                       ' Appel saisie
1800 PUT #1,CRANG
1810 CHASH%(CRANG)=CCLE:GOSUB 1590                  ' Appel sauvegarde de CHASH%
1820 GET #1,CRANG                                   ' Rappel client (indispensable)
1830 GOTO 1720
1840 '---------------------------------- SAISIE CLIENT/MODIFICATION
1850 PRINT "Rue:";TAB(15);RUE$;TAB(40);:INPUT X$:IF X$<>"" THEN LSET RUE$=X$
1860 PRINT "Ville:";TAB(15);VILLE$;TAB(40);:INPUT X$:IF X$<>"" THEN LSET VILLE$=X$
1870 PRINT "Code postal:";TAB(15);CPOST$;TAB(40);:INPUT X$:IF X$<>"" THEN LSET CPOS
T$=X$
1880 RETURN
1890 '=======================================================================
1900 '
```

```
10 ' gc  1.4.81
20 '
40 '         EDITION DE GRANDS CARACTERES  et GENERATEUR de PROGRAMMES
50 '
60 '   Ce programme permet:
70 '
80 '         1/ d'editer des caracteres geants
90 '         2/ de generer un programme qui editera ces caracteres geants
100 '
110 '
120 CLEAR(3000)
130 DIM AL$(26,5)                                  ' Table ALPHABET
140 DIM X(50)
150 '
160 DATA "       "
170 DATA "       "
180 DATA "       "
190 DATA "       "
200 DATA "       "
210 '
220 DATA "   *   "
230 DATA "  * *  "
240 DATA " ***** "
250 DATA "*     *"
260 DATA "*     *"
270 '
280 DATA "****** "
290 DATA "*     * "
300 DATA "*******"
310 DATA "*     *"
320 DATA "*******"
330 '
340 DATA "******"
350 DATA "*     "
360 DATA "*     "
370 DATA "*     "
380 DATA "******"

1490 '
1500 DATA "*     *"
1510 DATA "*     *"
1520 DATA " *   * "
1530 DATA "  * *  "
1540 DATA "   *   "
1580 '
1590 '
1600 '
1610 '
1620 '------------------------------ Lecture des DATAS dans AL$(,)
1630 '
1640 '      La table AL$(,) contient l'alphabet
1650 '
1660 FOR I=0 TO 22                    ' 22 lettres
1670   FOR J=1 TO 5                   ' 5 lignes par lettre
1680     READ AL$(I,J)
1690   NEXT J
1700 NEXT I
1710 '----------------------------------- MENU
1720 INPUT "MODE? (EGC,GP,..)  ";M$
1730 IF M$="EGC" THEN GOSUB 1770
1740 IF M$="GP" THEN GOSUB 1910
1750 GOTO 1720
1760 '==================================== EDITION D'UN MESSAGE
```

```
1764 '
1770 PRINT:INPUT "Message? ";M$
1780 FOR I=1 TO LEN(M$)
1790   X$=MID$(M$,I,1)
1800   X(I)=ASC(X$)-64                  ' X():pointeurs vers AL$(,)
1810   IF X(I)=-32 THEN X(I)=0
1820 NEXT I
1830 '
1840 FOR LIG=1 TO 5                     ' 5 lignes pour 1 caractere
1850   FOR J=1 TO LEN(M$)               ' Edition d'une ligne
1860     LPRINT AL$(X(J),LIG);"  ";
1870   NEXT J
1880 LPRINT
1890 NEXT LIG
1900 GOTO 1770
1910 '==================================== GENERATEUR DE PROGRAMMES
1920 PRINT:INPUT "Message? ";M$
1930 FOR I=1 TO LEN(M$)
1940   X$=MID$(M$,I,1)
1950   X(I)=ASC(X$)-64                  ' X():pointeurs vers AL$(,)
1960   IF X(I)=-32 THEN X(I)=0
1970 NEXT I
1980 INPUT "Nom programme genere? ";NP$
1990 OPEN "i",#1,NP$
2000 '
2010 FOR LIG=1 TO 5                     ' 5 lignes pour 1 caractere
2020   Y$=STR$(LIG)+" PRINT "+CHR$(34)
2030   FOR J=1 TO LEN(M$)
2040     Y$=Y$+AL$(X(J),LIG)+"  "
2050   NEXT J
2060   PRINT #1,Y$+CHR$(34)
2070   LPRINT Y$+CHR$(34)
2080 NEXT LIG
2090 CLOSE #1
2100 GOTO 1920
```

```
******  *****     *
*    *  *         *
******  ******    *
*            *    *
*       ******    *
```

```
 1 PRINT "******  *****     *     "
 2 PRINT "*    *  *         *     "
 3 PRINT "******  ******    *     "
 4 PRINT "*            *    *     "
 5 PRINT "*       ******    *     "
```

EDITION DE BULLETINS DE PAYE

Ce programme permet de *saisir les salaires du personnel, d'éditer des bulletins de paie* et d'obtenir différents totaux pour chaque catégorie de personnel (cadres, employés). Le récapitulatif général n'a pas été programmé.

Ce programme, relativement simple, devrait permettre par son accès facile d'observer certains aspects de l'organisation des programmes.

Tout d'abord, on a cherché à traiter les différentes catégories de personnel avec un même programme, les totaux pour les deux catégories étant stockés dans des tables de dimension 2. Un indice 'TYPE' que l'on positionne à 1 ou 2 permet d'accéder à l'élément du tableau cherché.

Trois sous-programmes ont été définis *(édition bulletin de paie, cumul, édition totaux)* non pas parce qu'ils pourraient être appelés de plusieurs endroits et ainsi permettre d'économiser de la place mémoire et de la programmation mais plutôt par *souci de clarté et de modularité.* Le *programme principal* sera plus court et n'en sera donc que plus lisible.

Il a été prévu à la fin de l'édition de chaque bulletin de paye la possibilité pour l'opérateur de refuser le cumul (en cas d'erreur de saisie par exemple).

PARTICULARITES DE PROGRAMMATION

Plutôt que de programmer :

```
PRINT USING "####.##";X
PRINT USING "####.##";Y
```

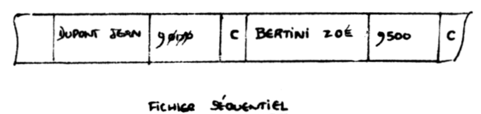
On aurait pu faire :

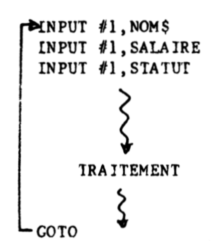
```
FORMAT$="####.##"
PRINT USING FORMAT$;x
PRINT USING FORMAT$;Y
```

L'introduction au clavier, à chaque paie, des noms du personnel ainsi que des salaires est fastidieuse. Si, de plus et pour une raison quelconque, il y a interruption du programme, toute l'introduction des données est à reprendre.

Une saisie du personnel pourrait être faite dans un fichier séquentiel qui serait relu par programme à chaque traitement.

```
OPEN FICHIER "PERSONNEL",No 1
```





Si on ne dispose pas de fichier mais qu'il reste de la place en mémoire centrale, les noms, salaires et statuts du personnel peuvent y être rangés par l'intermédiaire de DATA.

```
DATA DUPONT Jean,9000,C
DATA BERTINI Zoe,9500,C

READ NOM$
READ SALAIRE
READ STATUT

 TRAITEMENT

GOTO
```

PROGRAMME de PAYE
=================

```
        INPUT "Date ? ";DT$
              !
              !

DEBUT:  INPUT "Prix Repas ? "     ;PREPAS
        INPUT "Prix Transport ? ";PTRANSPORT
              !

MODE :  INPUT "Mode? ";MODE$
        IF MODE$="" GOTO DEBUT

        IF MODE$="CADRE"   THEN TYPE=1:GOTO NOM
        IF MODE$="EMPLOYE" THEN TYPE=2:GOTO NOM
        IF MODE$="TOTCAD"  THEN TYPE=1:GOSUB IMP-TOT
        IF MODE$="TOTEMP"  THEN TYPE=2:GOSUB IMP-TOT
        GOTO MODE

              !

NOM  :  INPUT "Nom? ";NOM$
        IF NOM$="" GOTO MODE

              !

        INPUT "Paye? ";PAYE
              !
        GOSUB IMP-PAYE

        INPUT "PAYE OK? ";RP$
        IF RP$><"O" goto NOM

        NB(TYPE)=NB(TYPE)+1

        GOSUB CUMUL

        GOTO NOM
```

SOUS-PROGRAMME IMPRESSION TOTAL … RETURN

IMPRESSION BULLETIN DE PAYE … RETURN

CUMUL … RETURN

```
10 '        Programme a usage PEDAGOGIQUE
20 '
30 PRINT "PAYE 28/7/80":PRINT ""
40 '
50          PLAFOND=5000
60          CRET(1,1)= .035                                 ' Maladie
70          CRET(2,1)= .010
80          CRET(3,1)= .047                                 ' Vieillesse
90          CRET(4,1)= .0084                                ' Chomage
100          FOR V=1 TO 4:CRET(V,2)=CRET(V,1):NEXT V
110          CRET(5,1)=.0618 :CRET(5,2 )=0                   ' Retraite Cadre
120          CRET(6,1)=.0245 :CRET(6,2)=CRET(6,1)            ' Retraite Complem
entaire
130 '
140 DATA S.S. MAL. S/SAL. TOTAL,S.S. MAL. S/SAL PLAF,S.S. VIEL S/SAL PLAF
150 DATA Assurance chomage,Retraite des cadres,Retraite complementaire
160 FOR I=1 TO 6:READ TYRET$(I):NEXT I
170 '
180 DATA CADRE,EMPLOYE:FOR I=1 TO 2:READ STATUT$(I):NEXT I
190 '
200 INPUT "Date? ";DATE$:IF DATE$="" GOTO 200
210 PRINT "":INPUT "Prix Repas ? ";PREPAS
220 INPUT "Transport  ? ";TRANSPORT
230 '------------------------------------------------------------ MENU
240 PRINT TAB(20);"Mode Emploi:":PRINT ""
250 PRINT TAB(20);"   C   : Bulletin Cadre"
260 PRINT TAB(20);"   E   : Bulletin Employe"
270 PRINT TAB(20);"   FC  : Total Cadre"
280 PRINT TAB(20);"   FE  : Total Employe"
290 PRINT TAB(20);"   Les Cadres et Employes peuvent etre Traites"
300 PRINT TAB(20);"   dans le desordre. FC et FE peuvent etre employes"
310 PRINT TAB(20);"   a n'importe quel moment"
320 PRINT "": INPUT "Cadre,Employe,Ouvrier? (C,E,FC,FE,FO,FIN) ";R$
330 '
340          IF R$="C" THEN TYPE=1:GOTO 410              ' Aiguillages
350          IF R$="E" THEN TYPE=2:GOTO 410
360          IF R$="FC" THEN  TYPE=1:GOSUB 1020:GOTO 320
370          IF R$="FE" THEN TYPE=2:GOSUB 1020:GOTO 320
380          IF R$="" GOTO 210
390 GOTO 240
400 '---------------------------------------------------------------------------
410 PRINT "":INPUT "Nom      ? ";NOM$                    ' Debut SAISIE
420 IF NOM$="" GOTO 240
430 INPUT "Prenom   ? ";PRENOM$
440 INPUT "Salaire  ? ";SALAIRE
450 INPUT "Qualif   ? ";QUALIF$
460 '
470          SBRUT=SALAIRE+PREPAS:X=SBRUT-PLAFOND:IF X<0 THEN X=0
480          Z=PLAFOND:IF SALAIRE<PLAFOND THEN Z=SBRUT
490          BRET(1)=SBRUT:BRET(2)=Z:BRET(3)=Z
500          BRET(4)=SBRUT:BRET(5)=X:BRET(6)=Z
510          R6=0:FOR V=1 TO 6:RET(V)=CRET(V,TYPE)*BRET(V):R6=R6+RET(V):NEXT V
520          SIMPOS=SBRUT-R6
530          SPAYE=SIMPO+TRANSPORT
540 GOSUB 830                                       ' Appel SPG impresion paye
550 '
560 PRINT "":INPUT "OK ? (O,N) ";R$:IF R$="N" GOTO 410
570 IF R$><"O" GOTO 560                            ' OK? peut on faire le CUMUL
580 GOSUB 650:GOTO 410
590 '
600 '-----------------------------------------------------------------------
```

```
640 '                                          SPG  Calcul CUMUL
650 TSALA(TYPE)=TSALA(TYPE)+SALAIRE
660 TREPA(TYPE)=TREPA(TYPE)+PREPAS
670 TBRUT(TYPE)=TBRUT(TYPE)+SBRUT
680 '
690  FOR V=1 TO 6
700       BASE(V,TYPE)=BASE(V,TYPE)+BRET(V)  :MRET(V,TYPE)=MRET(V,TYPE)+RET(V)
710  NEXT V
720 '
730 TENUE(TYPE)=TENUE(TYPE)+R6
740 TIMPO(TYPE)=TIMPO(TYPE)+SIMPO
750 TTRAN(TYPE)=TTRAN(TYPE)+TRANS
760 TNET(TYPE)=TNET(TYPE)+SPAYE
770 '
780 NB(TYPE)=NB(TYPE)+1
790 RETURN
800 '
810 '--------------------------------------------------------------------------
820                                            ' SPG Impression paye
830 LPRINT "":LPRINT DATE$,PRENOM$,NOM$,QUALIF$,STATUT$(TYPE):LPRINT "":LPRINT ""
840 LPRINT "Salaire Brut  " TAB(60); :LPRINT USING "#####.##";SBRUT
850 '
860 LPRINT "":LPRINT TAB(44) "Base" TAB(50)  "Retenues ":LPRINT ""
870 '
880  FOR V=1 TO 6
890       LPRINT TAB(6) TYRET$(V) TAB(34) :LPRINT USING ".####"; CRET(V,TYPE);
900       LPRINT TAB(40):LPRINT USING "#####.##";BRET(V);:LPRINT   TAB(50):LPRINT U
SING "#####.##";RET(V)
910  NEXT V
920 '
930 LPRINT "":LPRINT "Total des Retenues"TAB(60);:LPRINT USING "#####.##";R6
940 LPRINT "Salaire NET IMPOSABLE " TAB(60);:LPRINT USING "#####.##";SIMPO
950 LPRINT ""
960 LPRINT "Salaire a payer" TAB(60) :LPRINT USING "#####.##";SPAYE:LPRINT TAB(60)
"--------"
970 LPRINT ""
980 RETURN
990 '----------------------------------------------------------------------
1000 '
1010 '                                          ' SPG Impression Totaux
1020 LPRINT "":LPRINT "TOTAUX  "; STATUT$(TYPE);"      Nombre=";NB(TYPE):LPRINT ""
1030 LPRINT "Salaires de base" TAB(53)     :LPRINT USING "#######.##";TSALA(TYPE)
1040 LPRINT "Indeminites de REPAS" TAB(53) :LPRINT USING "#######.##";TREPA(TYPE)
1050 LPRINT "Salaires brut" TAB(53)        :LPRINT USING "#######.##";TBRUT(TYPE)
1060 LPRINT "":LPRINT TAB(37);"Base des retenues";TAB(55)"Retenues"::LPRINT ""
1070 '
1080  FOR V=1 TO 6
1090       LPRINT TAB(6) TYRET$(V) TAB(45) USING "######.##";BASE(V,TYPE);
1100       LPRINT USING "######.##";MRET(V,TYPE)
1110  NEXT V
1120 '
1130 LPRINT ""
1140 LPRINT "Total Retenues";TAB(53);:LPRINT USING "#######.##";TENUE(TYPE)
1150 LPRINT "":LPRINT  "Salaires NET IMPOSABLE " TAB(54) USING "######.##";TIMPO(TY
PE)
1160 '
1170 RETURN
1180 '
1190 '
```

```
1220 '
1230 '
1240 '    DEFINITION DES VARIABLES
1250 '    ========================
1260 '
1270 '    CRET          Table (6,2) des COEFFICIENTS de RETENUES
1280 '    TYRET$        Table (6)   des TYPES de RETENUES
1290 '    BRET          TABLE (6)   des BASES de RETENUES
1300 '    RET           TABLE (6)   des RETENUES
1310 '    R6            Total RETENUES
1320 '    X             Difference SALAIRE-PLAFOND SS
1330 '    Z             PLAFOND ou SALAIRE si SALAIRE<PLAFOND
1340 '    BASE          Table (6,2) des TOTAUX BASES
1350 '    MRET          Table (6,2) des Montants RETENUES
1360 '    TENUE         Table (2)   des totaux RETENUES par CATEGORIE
1370 '    TIMPO         Table (2)   des totaux IMPOSABLES par CATEGORIE
1380 '    TTRAN         Table (2)   des totaux TRANSPORTS par CATEGORIE
1390 '    TNET          Table (2)   des totaux SALAIRES NET par CATEGORIE
1400 '    TREPAS        Table (2)   des totaux REPAS par CATEGORIE
1410 '    TSALA         Table (2)   des totaux par CATEGORIE
1420 '
1430 '
1440 '
1450 '     Coefficients retenues    Bases retenues        Retenues
1460 '          CRET(,)                BRET()                RET()
1470 '      -----------------        ----------            ------
1480 '      ! .035   !  .035 !        !         !           !    !
1490 '      ! .010   !  .035 !        !         !           !    !
1500 '      ! .047   !  .047 !    X   !         !     =     !    !
1510 '      ! .0084  !  .0084!        !         !           !    !
1520 '      ! .0618  !  .0   !        !         !           !    !
1530 '      ! .0245  !  .0245!        !         !           !    !
1540 '      -----------------        ----------            ------
1550 '
1560 '      Cadres     Employes
1570 ' ----------------------------------------------------------------------
1580 '
1590 '      TOTAUX
1600 '
1610 '    Totaux base               Montants retenues
1620 '      BASE(,)                    MRET(,)
1630 '    -----------              ------------
1640 '  1 !    !    !              !     !     !
1650 '    !    !    !              !     !     !
1660 '    !    !    !              !     !     !
1670 '    !         !              !           !
1680 '    !         !              !           !
1690 '  6 !         !              !           !
1700 '    -----------              ------------
```

25.12.80	Nicole	LANGEARD	Secretaire	EMPLOYE

Salaire Brut				8200.00
		Base	Retenues	
S.S. MAL. S/SAL. TOTAL	.0350	8200.00	287.00	
S.S. MAL. S/SAL PLAF	.0100	5000.00	50.00	
S.S. VIEL S/SAL PLAF	.0470	5000.00	235.00	
Assurance chomage	.0084	8200.00	68.88	
Retraite des cadres	.0000	3200.00	0.00	
Retraite complementaire	.0245	5000.00	122.50	
Total des Retenues				763.38
Salaire NET IMPOSABLE				7436.62
Salaire a payer				7536.62

TOTAUX EMPLOYE Nombre= 3

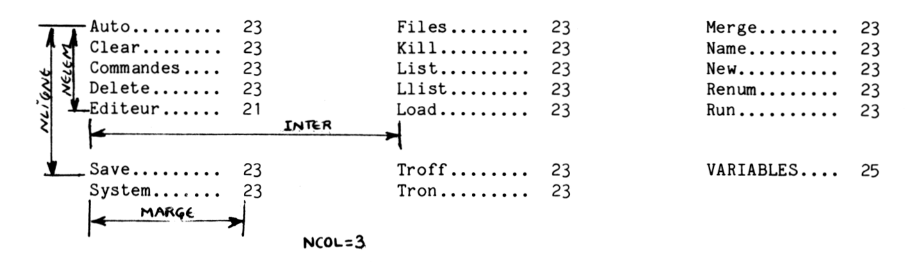
Salaires de base		24000.00
Indeminites de REPAS		600.00
Salaires brut		24600.00
	Base des retenues	Retenues
S.S. MAL. S/SAL. TOTAL	24600.00	861.00
S.S. MAL. S/SAL PLAF	15000.00	150.00
S.S. VIEL S/SAL PLAF	15000.00	705.00
Assurance chomage	24600.00	206.64
Retraite des cadres	9600.00	0.00
Retraite complementaire	15000.00	367.50
Total Retenues		2290.14
Salaires NET IMPOSABLE		22309.90

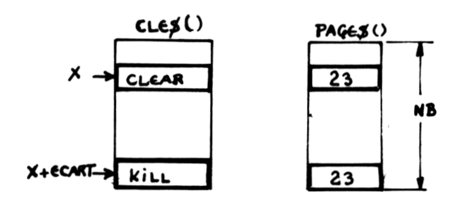
EDITION AUTOMATIQUE DE TABLEAUX

Ce programme permet d'éditer automatiquement sur plusieurs colonnes et sur plusieurs pages, deux tables à une dimension qui peuvent représenter par exemple un index de livre.

Sur l'exemple, les mots-clés et les numéros de pages sont fournis dans des DATA et lus dans deux tables CLE$() et PAGE$(). Ils pourraient tout aussi bien être extraits de fichiers à accès direct ou séquentiel.

Le principe consiste à imprimer sur une même ligne plusieurs éléments séparés par un écart égal au nombre de lignes par page.





```
30 ' EDITA              EDITION AUTOMATIQUE DE TABLEAU
40 '
50 '
60 '               (index de livre)
70 CLEAR(5000)
80 NB=20                         ' Nombre d'elements
90 DIM CLE$(NB),PAGE$(NB)
100 DATA Commandes,23
110 DATA Auto,23,Clear,23,Delete,23,Files,23
120 DATA Kill,23,List,23,Llist,23,Load,23,Name,23,New,23
130 DATA Merge,23,Renum,23,Run,23,Save,23,System,23,Tron,23
140 DATA Troff,23
150 DATA Editeur,21
160 DATA VARIABLES,25
170 '-------------------------------------------------------------------
180 '                             LECTURE DES DATAS dans CLE$() et PAGE$()
190 FOR I=1 TO NB
200   READ CLE$(I)
210   READ PAGE$(I)
220 NEXT I
230 '----------------------------------------------------------------------
240 K=NB                           ' TRI des TABLES
250 '
260 INV=0
270 FOR I=1 TO K-1
280   IF CLE$(I+1)<CLE$(I) THEN
        SWAP CLE$(I),CLE$(I+1):SWAP PAGE$(I),PAGE$(I+1):INV=1
290 NEXT I
310 IF INV=1 THEN K=K-1:GOTO 250
330 '-----------------------------------------------------------------------
340 '                              EDITION SUR 3 COLONNES
350 LPRINT
360 '
370 NLIGNE=6                   ' Nombre de lignes par page
380 NELEM=5                    ' Nombre de lignes a imprimer par page
390 NCOL=3                     ' Nombre de colonnes
400 INTER=30                   ' Intervalle entre 2 colonnes
410 MARGE=15                   ' Marge entre mot-cle et no de page
420 '
450 '
460 FOR PGE=1 TO 1000                                 ' 1000 pages
470   IF NB<NELEM*NCOL*(PGE-1) THEN STOP
480   ECART=INT((NB-(NELEM*NCOL)*(PGE-1)+NCOL-1)/NCOL) ' Page incomplete
490   IF NB>NELEM*NCOL*PGE THEN ECART=NELEM           ' Page complete
500 '
510    FOR LGNE=1 TO ECART                            ' Edition d'une page
520       X=LGNE+(PGE-1)*NELEM*NCOL                   ' X:element premiere colonne
530       FOR L=1 TO NCOL                             ' Edition d'une ligne
540          LPRINT TAB(INTER*(L-1));
550          IF X+ECART*(L-1)>NB THEN STOP
560          LPRINT CLE$(X+ECART*(L-1));STRING$(".",MARGE-2-LEN(CLE$(X+ECART*(L-1))
));
570          LPRINT TAB(MARGE+INTER*(L-1));
580          LPRINT PAGE$(X+ECART*(L-1));
590       NEXT L
600       LPRINT
610    NEXT LGNE
620 '
630 FOR K=NELEM TO NLIGNE:LPRINT:NEXT K
640 FOR K=2 TO 200:NEXT K
650 NEXT PGE
```

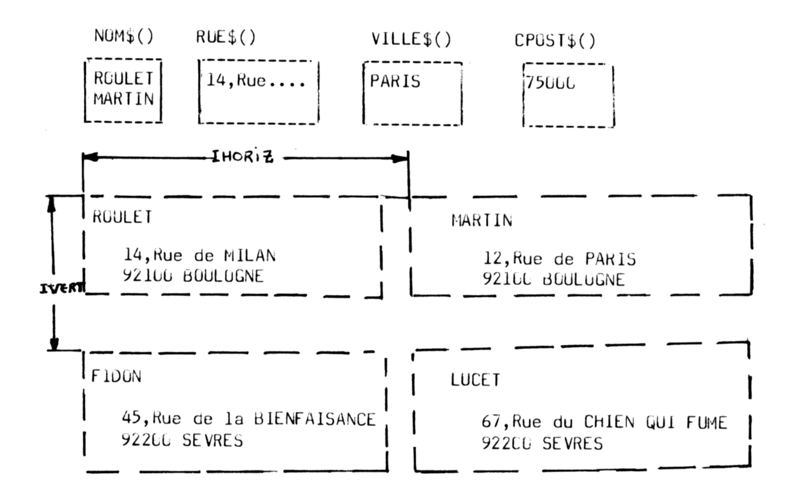
EDITION D'ETIQUETTES

Ce programme permet d'éditer des étiquettes d'adresses stockées dans un fichier Random.

Différentes variables définissent :

- le nombre d'étiquettes par ligne
- l'intervalle entre chaque étiquette

L'édition de plusieurs étiquettes sur la même ligne nous oblige à stocker les noms et adresses dans des tables avant de les imprimer.



BESSE

5,Rue la BRUYERE

```
30 ' ETIQ 3.11.80       EDITION D'ETIQUETTES
50 '
60 '
110 '
210 '
220 '   Si le saut de page n'est pas necessaire supprimer la ligne 630
230 '
240 OPEN "R",1,"ETIQ"
250 FIELD #1,15 AS NOM$,25 AS RUE$,15 AS VILLE$,6 AS CPOST$
260 '
270 INPUT "Mode? (CR,ETIQ,..) ";M$
280 IF M$="CR" THEN GOSUB 330
290 IF M$="ETIQ" THEN GOSUB 460
300 GOTO 270
310 '-------------------------------------------------------------------
320 '                              CREATION
330 NR=LOF(1)          ' Rangement en fin de fichier
340 GET #1,NE
350 INPUT "Nom? ";X$:IF X$<>"" THEN LSET NOM$=X$
360 IF X$="" THEN RETURN
370 LINE INPUT "Rue? ";X$:IF X$<>"" THEN LSET RUE$=X$
380 INPUT "Ville? ";X$:IF X$<>"" THEN LSET VILLE$=X$
390 INPUT "Cpost? ";X$:IF X$<>"" THEN LSET CPOST$=X$
400 PUT #1,NR
410 PRINT"RANGE EN:";NR
420 GOTO 330
430 '===================================================================
440 '                            SORTIE ETIQUETTES
450 '
460 NE=2          ' Nombre etiquettes par ligne
470 IHORIZ=35     ' Intervalle horizontal
480 IVERT=8       ' Intervalle vertical
490 NLPAGE=71     ' Nombre de lignes par page
500 MARGE=2         ' Marge de debut
510 '
520 NRANG=INT((NLP)/IVERT)              ' Nombre de rangees par page
530 SP=NLPAGE-NRANG*IVERT                     ' Saut de page
540 '
550 N=0:TRANG=0
560 '
570 FOR I=1 TO LOF(1)             ' Lecture de tout le fichier
580   GET #1,I
590   IF ASC(NOM$)=0 THEN 640     ' Enregistrement vide?
600   N=N+1
610   NOM$(N)=NOM$:RUE$(N)=RUE$:CPOST$(N)=CPOST$:VILLE$(N)=VILLE$
620   IF N=NE THEN GOSUB 700:TRANG=TRANG+1: N=0
630   IF TRANG=NRANG THEN FOR K=1 TO SP:LPRINT:NEXT K:TRANG=0      ' Saut de page
640 NEXT I
650 IF N>=1 THEN GOSUB 700
660 RETURN
670 '-------------------------------------
680 '    SPGM Edition etiquettes
690 '
700 FOR K=1 TO N:LPRINT TAB(MARGE+(K-1)*IHORIZ);NOM$(K);:NEXT K:LPRINT:LPRINT
710 FOR K=1 TO N:LPRINT TAB(MARGE+3+(K-1)*IHORIZ);RUE$(K);:NEXT K:LPRINT
720 FOR K=1 TO N:LPRINT TAB(MARGE+3+(K-1)*IHORIZ);CPOST$;VILLE$;:NEXT K:LPRINT
730 '
740 FOR K=1 TO IVERT-4:LPRINT:NEXT K       ' Intervalle vertical (4=nb lignes impri
mees avant)
750 RETURN
```

INITIATION AU TRAITEMENT DE TEXTE

Le traitement de texte en Basic peut être fait en lisant un fichier texte, caractère par caractère, (à l'aide de INPUT$#), mais en Basic interprété, le temps d'analyse de chaque caractère est relativement long.

L'utilisation de la fonction **INSTR** (début, chaîne à analyser, chaîne cherchée), plus rapide et plus pratique, présente cependant la restriction suivante :

Il faut prendre soin, si par exemple on recherche la première chaîne 'FOR' placée entre guillemets dans une ligne 'BASIC', de limiter le domaine de recherche de 'FOR' en ayant auparavant recherché les positions des deux premiers guillemets.

```
---" FOR "-------!------ FOR ------
```

On recherche le FOR entre guillemets.

Dans ce programme de décalages de boucles FOR-NEXT il s'agit, à chaque fois qu'un FOR est rencontré, d'augmenter la marge d'impression de trois par exemple. Au contraire, pour chaque NEXT rencontré, on diminue la marge de trois.

Si les FOR-NEXT ne sont pas appaires, (ils doivent l'être avec les Basics compilés), nous nous recadrons sur la première boucle 'FOR' externe. Naturellement, les FOR, NEXT, REM entre guillemets ne doivent pas être pris en considération.

Notons au passage que si les Basics interprétés acceptent les FOR-NEXT non appaires, il est vivement conseillé, pour des raisons de lisibilité, de ne pas utiliser cette facilité. Le nom de variable doit aussi de préférence être noté après NEXT, non seulement pour la lisibilité mais pour éviter qu'un programme se poursuive en incrémentant une autre variable que celle qui a été implicitement prévue par le programmeur.

Détails du programme :

- On lit le programme Basic ligne par ligne
- La partie programme de chaque ligne est séparée des commentaires (recherche de ' et REM)
- On recherche ensuite les FOR et NEXT

```
10 ' DFOR 30.10.80
20 '
30 '      PROGRAMME DE DECALAGE DE BOUCLES 'FOR'
40 '
50 '     Ce programme permet d'editer un programme en cadrant les boucles FOR
60 '     et les commentaires.
70 '     Le programme traite doit etre sauvegarde sous forme ASCII(i.e.non
80 '     preinterpretee ) par ' SAVE "XXX",A '
90 '
100 '
110 '    CADRAGE des COMMENTAIRES en COLONNE 60
120 '
130 '       VERSIONS MICROSOFT AVEC ESPACES SEPARATEURS(A ADAPTER POUR TRS80)_
140 '
150 '      LIGNE$  : Ligne lue dans le fichier
160 '      LTRAIT$ : Ligne sans les commentaires    ( ' ou :REM)
170 '
180 MARGE=5
190 LIGNE$=""
200 INPUT "NOM DU PROGRAMME? ";PGM$    ' Sauvegarde en ASCII
210 OPEN "I",#1,PGM$
220 '
230 IF EOF(1)=-1 THEN STOP            ' Fin de fichier
240 INPUT #1,NUML                     ' No de ligne
250 LPRINT NUML;
260 '---------------------------------------------------------------------------
270 '                          separation pgm et com
280 '
290 '    -----"    '   "------ "  '   "---------- '  XXXXXX   Analyse commentaire
300 '         !    !   !
310 '         p1   pe  p2
320 CC=50     ' Cadrage commentaire
330 '
340 LINE INPUT #1,LIGNE$:LIGNE$=" "+LIGNE$  ' Lecture d'un enregistrement
350 '
360 PE=0:P1=0
370 '
380 P1=INSTR(PE+1,LIGNE$,CHR$(34)):IF P1<>0 THEN P2=INSTR(P1+1,LIGNE$,CHR$(34)) ELS.
E P2=0
390 '
400 P(1)=INSTR(PE+1,LIGNE$,"'"):
410 P(2)=INSTR(PE+1,LIGNE$," REM")                      ' "REM" sur TRS80
420 P(3)=INSTR(PE+1,LIGNE$,":REM")
430 '
440 PE=255
450 FOR I=1 TO 3    ' Recherche du plus petit
460   IF P(I)<>0 THEN IF P(I)<PE THEN PE=P(I):RP=I
470 NEXT I
480 '
490 IF PE=255 THEN PE=0
500 '
510 IF PE<>0 THEN IF PE>P1 AND PE<P2 THEN PE=P2:GOTO 380
520 IF PE<>0 THEN LTRAIT$=LEFT$(LIGNE$,PE-1):COM$=RIGHT$(LIGNE$,LEN(LIGNE$)-PE+1) E
LSE LTRAIT$=LIGNE$:COM$=""
521 '
522 '
523 '
524 '
525 '
526 '
527 '
530 '
```

```
531 '
532 '
533 '
534 '
538 '------------------------------------------------------------------------
540 '                                         Recherche FOR et NEXT
550 '
560 '    ---"    FOR   "-------"    FOR "-----  FOR I=1 TO 10   Analyse d'une ligne
570 '       !     !      !
580 '       P1    pe     P2
590 '
600 '   FC : Limite de la chaine a analyser   P1 : Premiere quote  P2 : Deuxieme qu
ote
610 '   FR : Presence de FOR dans la ligne    NX : Presence de NEXT
620 '
630 PE=0:PE=0:FR=0:NX=0:AMARGE=MARGE
640 '
650 P1=INSTR(PE+1,LTRAIT$,CHR$(34)):IF P1<>0 THEN P2=INSTR(P1+1,LTRAIT$,CHR$(34)):
ELSE P2=0
660 FC=P2:IF FC=0 THEN FC=255
670 '
680 X$=LEFT$(LTRAIT$,FC)
690 P(1)=INSTR(PE+1,X$," FOR ")                        ' "FOR" sur TRS80
700 P(2)=INSTR(PE+1,X$,":FOR")
710 P(3)=INSTR(PE+1,X$," NEXT ")                       ' "NEXT" sur TRS80
720 P(4)=INSTR(PE+1,X$,":NEXT ")
730 PE=255
740 FOR I=1 TO 4                               ' Recherche du plus pres
750   IF P(I)<>0 THEN IF P(I)<PE THEN PE=P(I):RP=I
760 NEXT I
770 '
780 IF PE=255 THEN PE=0
790 '
800 IF PE<>0 THEN IF RP<3 THEN IF PE>P1 AND PE<P2 THEN PE=P2 ELSE MARGE=MARGE+3::FR
=1
810 IF PE<>0 THEN IF RP>2 THEN IF PE>P1 AND PE<P2 THEN PE=P2 ELSE GOSUB 970:NX=1
820 IF PE<>0 THEN GOTO 650
830 IF FC<255 THEN PE=P2:GOTO 650
840 '---------
850 IF FR=1 THEN LPRINT TAB(AMARGE);LTRAIT$;TAB(CC);COM$:GOTO 230
860 IF NX=1 THEN LPRINT TAB(MARGE);LTRAIT$;TAB(CC);COM$:GOTO 230
870 '------------------------------------------------------------
880 '
890 IF LTRAIT$="" THEN LPRINT ;COM$:GOTO 230    ' Ligne commentaire
900 IF LTRAIT$=" " THEN LPRINT " ";COM$:GOTO 230  ' Ligne commentaire
910 '
920 IF LTRAIT$<>" " THEN LPRINT TAB(MARGE);LTRAIT$;TAB(CC);COM$:GOTO 230
930 '
940 '----------------------------------------------------------
950 '       Traitement de NEXT ou NEXT I,J,K
960 '
970 IF MARGE>5 THEN MARGE=MARGE-3 ELSE PRINT "FOR-NEXT PAS APPAIRES EN:";NUML
980 PP=INSTR(PE+1,LTRAIT$,":"):IF PP=0 THEN LNEXT$=LTRAIT$ ELSE LNEXT$=LEFT$(LTRAIT
$,PP)
990 PV=PE
1000 PV=INSTR(PV+1,LNEXT$,","):IF PV=0 THEN 1020 ELSE IF MARGE>5 THEN MARGE=MARGE-3
:GOTO 1000 ELSE PRINT "FOR-NEXT PAS APPAIRES EN:";NUML:GOTO 1000
1010 '
1020 RETURN
1030 '-----------------------------------------------------------------------
1060 '
```

AVANT

```
10 '  ESSAI DE DECALAGES DE BOUCLES 'FOR'
20 '
30 FOR I=1 TO 10 ' POUR I=1 TO 10
40 PRINT I;
50 NEXT I     ' I=I+1
60 '
70 FOR I=1 TO 5
80 FOR J=1 TO 10
90 PRINT "I=";I;"J=";J
95 PRINT " FOR: ";" NEXT ";
100 NEXT J
110 NEXT I
120 '
130 FOR I=1 TO 5:FOR J=1 TO 3   ' Essai avec NEXT I,J
140 PRINT "FOR "
150 NEXT I,J
160 '
170 FOR P=1 TO 5  ' FOR-NEXT non appaires.
180 IF P=3 THEN NEXT P
190 NEXT P
200 '
210 FOR I=1 TO 5  ' Recadrage sur boucle externe
220 PRINT I
230 NEXT I
240 '-----------------------------------------------------------------
```

APRES

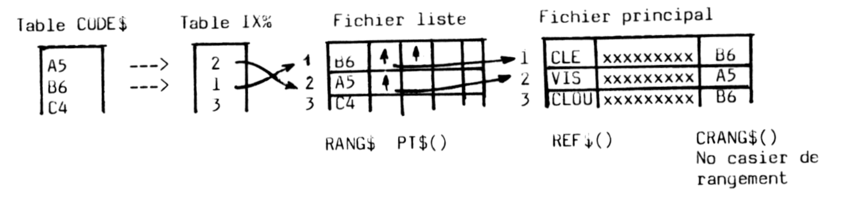
```
 10   '  ESSAI DE DECALAGES DE BOUCLES 'FOR'
 20   '
 30   FOR I=1 TO 10                              ' POUR I=1 TO 10
 40      PRINT I;
 50   NEXT I                                     ' I=I+1
 60   '
 70   FOR I=1 TO 5
 80      FOR J=1 TO 10
 90          PRINT "I=";I;"J=";J
 95          PRINT " FOR: ";" NEXT ";
 100     NEXT J
 110  NEXT I
 120  '
 130  FOR I=1 TO 5:FOR J=1 TO 3                  ' Essai avec NEXT I,J
 140         PRINT "FOR "
 150  NEXT I,J
 160  '
 170  FOR P=1 TO 5                               ' FOR-NEXT non appaires.
 180  IF P=3 THEN NEXT P
 190  NEXT P
 200  '
 210  FOR I=1 TO 5                               ' Recadrage sur boucle externe
 220     PRINT I
 230  NEXT I
 240  '-----------------------------------------------------------------
```

LISTES INVERSES

Les systèmes de fichiers actuels ne permettent pas d'accéder aux enregistrements par leur contenu (mémoires associatives). Par exemple, il faut lorsque l'on recherche dans un fichier toutes les pièces d'un casier de rangement, explorer le fichier des stocks dans sa totalité en ne sélectant que les enregistrements correspondant au casier concerné.

Ceci peut devenir long pour des fichiers de taille importante. Aussi est-on amené à créer une liste inverse donnant pour chaque casier, la liste de tous les enregistrements qui lui sont relatifs.

Les listes inverses sont mises à jour en 'temps réel' à chaque ajout dans le fichier principal. Il est cependant prévu de les recréer en différé en cas d'incident.

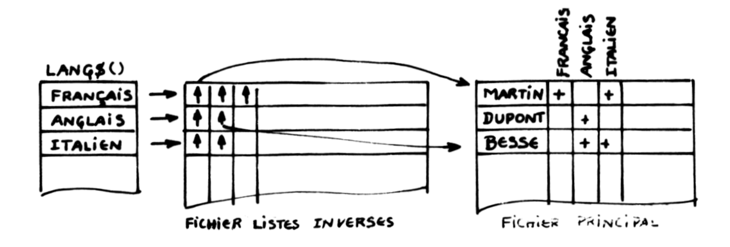


Exemple d'exécution :

Quel casier? B6

CLE
CLOU

Si les mots-clés sont déterminés et fixés, une organisation plus économique est possible. Sur cet exemple, une simple bit-map permet de coder pour chaque personne, les différentes langues parlées.



```
10 ' LINVS 8.1.81
20 '                    LISTE INVERSEE
30 '       REF$    : REFERENCE              #1
40 '       CRANG$ : CASIER RANGEMENT        #1
50 '       RANG$  : CASIER RANGEMENT        #2
60 '       CODE$  : TABLE DES CODES (CASIERS)
70 '       IX%    : INDEX VERS LE FICHIER INDEX (#2)
80 '       PT$    : POINTEURS VERS LE FICHIER PRINCIPAL
90 '
100 DIM PT$(110),CODE$(50),IX%(50)
110 OPEN "R",1,"LINVS"          ' Fichier principal
120 FIELD #1,12 AS REF$,30 AS LIB$,4 AS PACHA$,4 AS PVENTE$,
     4 AS QV$,4 AS STK$,5 AS CRANG$
130 OPEN "R",#2,"ILINVS"        ' Fichier listes inverses
140 FOR I=1 TO 110:FIELD #2,5 AS RANG$,(I-1)*2 AS D$,2 AS PT$(I):NEXT I
150 GOSUB 920:GOSUB 680         ' APPEL CONSTITUTION CLE$(),IX%()
160 '============================================================
170 CLS:PRINT TAB(10) "PROGRAMME DE LISTE INVERSE":PRINT    ' MENU
180 PRINT TAB(10) "MODES:":PRINT
190 PRINT TAB(20) "C     :CREATION"
200 PRINT TAB(20) "CRI   :CREATION INDEX"
210 PRINT TAB(20) "LISTE :LISTE PAR CASIER"
220 PRINT::INPUT "MODE (C,LISTE,CRI,RCAS ";MODE$
230 IF MODE$="C" THEN GOSUB 290
240 IF MODE$="CRI" THEN GOSUB 380:GOSUB 680
250 IF MODE$="LISTE" THEN GOSUB 760
260 IF MODE$="RCAS" THEN GOSUB 1020
270 GOTO 220
280 '============================================================
290 NE=LOF(1)+1:GET #1,NE                     ' CREATION (C)
300 X$="":INPUT "REFERENCE ";X$:IF X$="" THEN RETURN
310 Y$="":INPUT "RANGEMENT ";Y$
320 LSET REF$=X$:LSET CRANG$=Y$
330 PUT #1,NE
340 GOSUB 480                               ' APPEL MAJ LISTE INVERSE
350 GOTO 290
360 '============================================================
370 '                          CREATION LISTE INVERSE(CRI)
380 CLOSE #2:KILL "ILINVS":OPEN "R",#2,"ILINVS"
390 FOR I=1 TO 50:CODE$(I)="":NEXT I:NB=0    ' NB:NOMBRE DE CLES
400 FOR NE=1 TO LOF(1)
410    GET#1,NE:IF ASC(REF$)=0 GOTO 440
420    PRINT REF$,CRANG$
430    GOSUB 480
440 NEXT NE
450 CLOSE #2:OPEN "R",#2,"ILINVS"
460 RETURN
470 '-------------------------------- MAJ LISTE INVERSE
480    FOR K=1 TO 50
490      IF CRANG$=CODE$(K) GOTO 540       ' LE CODE EXISTE DEJA
500      IF CODE$(K)="" THEN CODE$(K)=CRANG$:IX%(K)=K:NB=NB+1:GOTO 540
510    NEXT K
520    STOP
530 '
540   GET #2,IX%(K)
550   FOR I=1 TO 110
560     IF CVI(PT$(I))=0 THEN LSET PT$(I)=MKI$(NE):
          LSET RANG$=CODE$(K):PUT #2,K:GOTO 590
570   NEXT I
580   STOP
590   RETURN
600 '============================================================
```

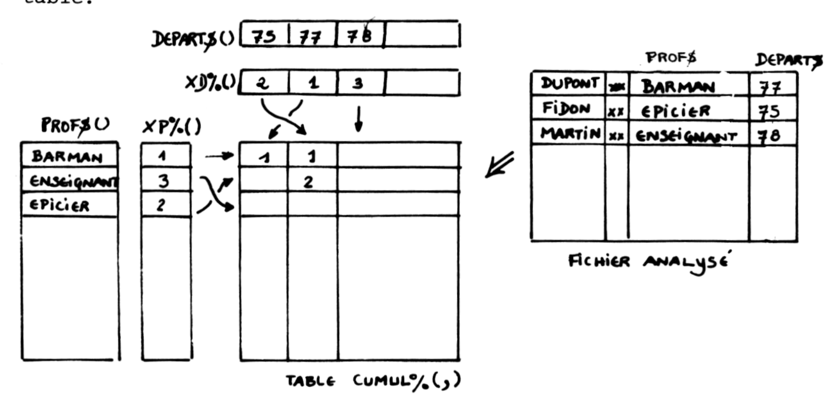
```
610 '
620 '
630 '
640 '
650 '
660 '
670 '-----------------------------------------------
680 FOR I=1 TO NB-1                         ' TRI  DE CLE$() ET IX%()
690    FOR J=I+1 TO NB
700        IF CODE$(J)<CODE$(I) THEN
               X$=CODE$(J):CODE$(J)=CODE$(I):CODE$(I)=X$:
               X=IX%(I):IX%(I)=IX%(J):IX%(J)=X
710     NEXT J
720 NEXT I
730 RETURN
740 '============================================================
750 '                           LISTE PAR CASIER(LISTE)
760 PRINT :PRINT "CASIER" TAB(20) "REFERENCE":PRINT
770 FOR CODE=1 TO LOF(2)
780   GET #2,IX%(CODE)
790   IF ASC(RANG$)=0 GOTO 880
800   PRINT RANG$;
810 '
820    FOR I=1 TO 55
830      NE=CVI(PT$(I)):IF NE=0 GOTO 870
840      GET #1,NE:PRINT TAB(20) REF$
850    NEXT I
860 '
870    PRINT
880 NEXT CODE
890 RETURN
900 '============================================================
910 '               CONSTITUTION DES TABLES CODE$ ET IX%
920 NB=0                              ' NB:NOMBRE DE CLES
930 FOR I=1 TO LOF(2)
940   GET #2,I
950   IF ASC(RANG$)=0 THEN GOTO 980
960     PRINT RANG$,I
970   NB=NB+1:CODE$(NB)=RANG$:IX%(NB)=I
980 NEXT I
990 RETURN
1000 '============================================================
1010 '                          INTERROGATION PAR CASIER(RCAS)
1020 PRINT:X$="":INPUT "CASIER CHERCHE ";X$:IF X$="" THEN RETURN
1030 FOR I=1 TO NB
1040   IF X$=LEFT$(CODE$(I),LEN(X$)) THEN GOTO 1090
1050 NEXT I
1060 '
1070 PRINT "CE CASIER N'EXISTE PAS":PRINT:GOTO 1020
1080 '
1090 GET #2,IX%(I):PRINT
1100 FOR K=1 TO 55
1110   NE=CVI(PT$(K)):IF NE=0 GOTO 1140
1120   GET #1,NE
1130   PRINT REF$
1140 NEXT K
1150 GOTO 1020
```

ANALYSE D'UN FICHIER SELON DEUX CRITERES

Nous analysons, dans un fichier d'adresses, la répartition des professions par département. Pour cela, nous remplissons une table à deux dimensions CUMUL%(,).

Afin d'éditer les résultats dans l'ordre croissant des départements et des professions, nous utilisons deux couples de tables DEPART$-XD% et PROF$-XP% qui seront triées avant l'édition de CUMUL% (voir dessin).

Pour chaque enregistrement lu dans le fichier, nous recherchons si le département et la profession existe déjà dans DEPART$() et PROF$(). S'ils n'existent pas, nous les ajoutons en fin de table.



```
10 ' ANA 12.10.80
20 '
30 '    ANALYSE D'UN FICHIER SELON 2 CRITERES
40 '    ===================================
50 '
60 '   DP    : departement
70 '   PF    : profession
80 '   NPROF : nombre de professions
90 '   NDEP  : nombre de departements
100 '
110 CLEAR(2000)
120 MDEP=20            ' Nombre de departements maxi
130 MPROF=20           ' Nombre de profession max
140 '
150 DIM DEPART$(MDEP),PROF$(MDEP)
160 '
170 NDEP=0:NPROF=0
180 '
190 OPEN "R",#1,"ANA"
200 FIELD #1,12 AS NOM$,10 AS DEPART$,10 AS PROF$
210 '
220 INPUT "MODE? ";M$
230 IF M$="CR" THEN GOSUB 270
240 IF M$="LIST" THEN GOSUB 370
250 GOTO 220
260 '-------------------------------------------------- CREATION
270 ARANG=LOF(1)               ' LOF(1)+1 sur TRS80
280 GET #1,ARANG
290 INPUT "NOM? ";X$:IF X$="" THEN RETURN ELSE LSET NOM$=X$
300 INPUT "DEPARTEMENT ? ";X$:LSET DEPART$=X$
310 INPUT "PROFESSION? ";X$:LSET PROF$=X$
320 PUT #1,ARANG
330 PRINT:PRINT"RANGE EN:";ARANG
340 GOTO 270
350 '----------------------------------------------------------
360 '                                        LECTURE DU FICHIER
370 FOR NE=1 TO LOF(1)
380   GET #1,NE:IF ASC(NOM$)=0 THEN 530
390   PRINT NOM$,DEPART$,PROF$
400 '
410   FOR DP=1 TO MDEP          ' Recherche departement
420     IF DEPART$=DEPART$(DP) THEN GOTO 460
430     IF DEPART$(DP)="" THEN
         DEPART$(DP)=DEPART$:XD%(DP)=DP:NDEP=NDEP+1:GOTO 460
440   NEXT DP
450 '
460   FOR PF=1 TO MPROF         ' Recherche profession
470     IF PROF$=PROF$(PF) THEN GOTO 510
480     IF PROF$(PF)="" THEN
          PROF$(PF)=PROF$:XP%(PF)=PF:NPROF=NPROF+1:GOTO 510
490   NEXT PF
500 '
510 CUMUL%(DP,PF)=CUMUL%(DP,PF)+1
520 '
530 NEXT NE
540 '
541 '
542 '
550 '-----------------------------------------------------------
551 '
```

```
559 '
560 '                          TRI DES DEPARTEMENTS (bubble)
570 FOR I=1 TO NDEP-1
580   FOR J=I+1 TO NDEP
590     IF DEPART$(J)<DEPART$(I) THEN
          SWAP DEPART$(I),DEPART$(J):SWAP XD%(I),XD%(J)
600   NEXT J
610 NEXT I
620 '-----------------------------------------------------------
630 '                                     TRI DES PROFESSIONS (bubble)
640 FOR I=1 TO NPROF-1
650   FOR J=I+1 TO NPROF
660     IF PROF$(J)<PROF$(I) THEN
           SWAP PROF$(I),PROF$(J):SWAP XP%(I),XP%(J)
670   NEXT J
680 NEXT I
690 '=================================================================
700 '                                   EDITION DES RESULTATS
710 LPRINT:LPRINT
720 FOR L=1 TO 5          ' Edition des departements (en vertical)
730   LPRINT TAB(10+4);
740   FOR DP=1 TO NDEP
750     X$=MID$(DEPART$(DP),L,1):LPRINT X$;SPC(4);
760   NEXT DP
770   LPRINT
780 NEXT L
790 '--------------------------------------------------------------
800 FOR PF=1 TO NPROF                    ' Edition de la table CUMUL%
810   LPRINT  PROF$(PF);TAB(10)
820   TPROF=0
830   FOR DP=1 TO NDEP
840      X=CUMUL%(XD%(DP),XP%(PF))
850     IF X><0 THEN LPRINT USING "#####";X;       ELSE LPRINT "    .";
860     TPROF=TPROF+X
870     TTAL(DP)=TTAL(DP)+X
880   NEXT DP
890   LPRINT USING "#####";TPROF
900 NEXT PF
910 '---------------------------------------------------------
920 LPRINT:LPRINT TAB(10);
930 FOR DP=1 TO NDEP
940   LPRINT USING "#####";TTAL(DP);
950 NEXT DP
960 RETURN
```

```
              7    7    7    9
              5    7    8    2


BARMAN        .    1    .    .    1
ENSEIGNANT    .    .    1    .    1
EPICIER       2    .    .    .    2
PLOMBIER      .    .    .    1    1
POMPISTE      1    .    .    .    1

              3    1    1    1
```

INTERROGATION DE FICHIER

Plutôt que d'écrire des programmes d'interrogation de fichier pour chaque type de demande, il est intéressant de disposer d'un langage d'interrogation général simple à utiliser.

Nous allons d'abord traiter un cas simple de recherche multi-critères du type :

'Quels sont tous les plombiers de paris gagnant plus de 7000F'

L'opérateur formulera sa question sous la forme suivante :

PROF=PLOMBIER*VILLE=PARIS*SALAIRE>7000

Il nous faut définir les noms des zones du fichier. Nous le faisons par une table NZ$() contenant *NOM, PR, PROF, VILLE, SALAIRE*

Une fois l'interrogation formulée par l'opérateur, nous l'analysons pour constituer une table CRIT$() contenant les critères recherchés : PLOMBIER,PARIS et >7000 dans l'exemple choisi. Après avoir constitué cette table, nous documentons, pour chaque enregistrement lu dans le fichier, une table de vérité TV() avec 1 si le critère est vérifié ou 0 si le critère n'est pas vérifié. Ensuite, nous effectuons un 'ET' logique sur cette table de vérité.

Si le résultat est égal à 1, nous sélectons l'enregistrement lu. Pour un fichier de taille importante, la sélection devient longue. Aussi est-on amené, pour des fichiers importants, à définir au moment de la création des enregistrements des couples **mots-clés, pointeurs** permettant de retrouver directement les enregistrements concernés par l'interrogation.

Pour une recherche plus élaborée qu'un simple *'ET logique'* comme ci-dessus, nous pouvons formuler une interrogation de la façon suivante :

(PROF=PLOMBIER+PROF=EPICIER)*VILLE=PARIS

Il nous faut alors évaluer l'expression par une méthode que nous verrons plus loin.

```
10 ' INTF 31.10.80
20 '
30 '       INTERROGATION DE FICHIER
40 '
50 '
60 '   Ce programme permet de SELECTER les enregistrements repondant a un ou
70 '   plusieurs criteres, e.g.  'Quels sont tous les PLOMBIERS de PARIS gagnant
80 '   plus de 7000f
90 '
100 '        PROF=PLOMBIER*VILLE=PARIS*SALAIRE>7000
110 '
120 '                Fichier 'NOM'
130 '
135 '    ZN$(1)
140 '    ------------------------------------
150 '    !NOM ! PR ! VILLE ! PROF ! SALAIRE !
160 '    ------------------------------------
170 '    !    !    !       !      !         !
180 '    ------------------------------------
190 '
200 NZ=5
210 DATA NOM,PR,VILLE,PROF,SALAIRE
220 FOR I=1 TO NZ:READ NZ$(I):NEXT I       ' Table des noms de zone
230 '
240 DATA 15,12,20,15,4
250 FOR I=1 TO NZ:READ LZ(I):NEXT I       ' Table des longueurs des zones fichier
260 '
270 DATA 1,1,1,1,2:FOR I=1 TO NZ:READ TZ(I):NEXT I  ' Type de zone 1 ou 2
280 '
290 OPEN "R",#1,"INTF"
300 S=0:FOR I=1 TO NZ:FIELD #1,S AS D$,LZ(I) AS ZN$(I):S=S+LZ(I):NEXT I
310 '-------------------------------------------------------------------
320 '                                                    MENU
330 '
340 PRINT:INPUT "Mode? (CR,ET,...) ";M$
350 IF M$="CR" THEN GOSUB 400
360 IF M$="ET" THEN GOSUB 630
370 GOTO 340
380 '-------------------------------------------------------------------
390 '                          CREATION D'ENREGISTREMENTS
400 PRINT
410 NE=LOF(1)   ' Ajout en fin de fichier (LOF(1)+1 sur TRS80)
420 GET #1,NE
430 FOR I=1 TO NZ
440   PRINT NZ$(I);TAB(20);:INPUT X$
450   IF TZ(I)=1 THEN LSET ZN$(I)=X$ ELSE LSET ZN$(I)=MKS$(VAL(X$))
460 NEXT I
470 PUT #1,NE
480 PRINT: PRINT "Range en:";NE
490 RETURN
500 '=====================================================================
510 '                      INTERROGATION par 'ET'
520 '
530 '   NZ$    OPE$     CRIT$     TV
540 '  ------  -----   -------    ---
550 '  !NOM !  !XXX!   !XXXXX!    ! !
560 '  !PR  !  !XXX!   !XXXXX!    ! !
570 '  !VILL!  ! = !   !VERSA!    !1!
580 '  !PROF!  ! = !   !PLOMB!    !0!
590 '  !SALA!  ! > !   !5000 !    !1!
600 '  ------  -----   -------    ---
610 '
```

```
620 '
630 PRINT:FOR I=1 TO NZ:PRINT NZ$(I);"  ";:NEXT I:PRINT
640 PRINT:INPUT "EXpression : Nom zone=xxxx*Nom zone=*Nom zone>999 ";X$
650 GOSUB 890
660 '
670 PRINT
680 FOR NE=1 TO LOF(1)        ' Lecture de tout le fichier
690   GET #1,NE
700   IF ASC(ZN$(1))=0 THEN 820
710   FOR K=1 TO NZ:TV(K)=0:NEXT K
720   FOR K=1 TO NZ               ' Constitution de table de verite TV()
730     IF OPE$(K)="" THEN 760
740     IF OPE$(K)="=" THEN IF CRIT$(K)=LEFT$(ZN$(K),LEN(CRIT$(K))) THEN TV(K)=1
750     IF OPE$(K)=">" THEN IF CVS(ZN$(K))>VAL(CRIT$(K)) THEN TV(K)=1
760   NEXT K
770   RESUL=1
780   FOR K=1 TO NZ               ' Test table de verite TV()
790      IF OPE$(K)<>"" THEN RESUL=RESUL*TV(K)
800   NEXT K
810   IF RESUL=1 THEN PRINT ZN$(1);ZN$(2);ZN$(3);ZN$(4);CVS(ZN$(5))
820 NEXT NE
830 RETURN
840 '-----------------------------------------------------------------
850 '        Analyse de l'expression X$
860 '
870 '   SPGM   Constitution table OPE$()   et CRIT$()
880 '
890 FOR I=1 TO NZ:OPE$(I)="":CRIT$(I)="":NEXT I
900 CUR=1
910 '
920 DEP=CUR
930 '
940 Y$=MID$(X$,CUR,1)
950 '
960 IF Y$="=" THEN GOSUB 1040:CUR=CUR+1:GOTO 920
970 IF Y$=">" THEN GOSUB 1040:CUR=CUR+1:GOTO 920
980 IF Y$="*" THEN GOSUB 1110:CUR=CUR+1:GOTO 920
990 IF Y$="" THEN GOSUB 1110:RETURN
1000 CUR=CUR+1
1010 GOTO 940
1020 '--------------------------------------------
1030 '        Recherche nom de zone
1040 NZ$=MID$(X$,DEP,CUR-DEP)
1050 FOR I=1 TO NZ
1060   IF NZ$=NZ$(I) THEN OPE$(I)=Y$:PT=I:RETURN
1070 NEXT I
1080 RETURN
1090 '------------------------------------------
1100 '        Recherche critere
1110 CRIT$=MID$(X$,DEP,CUR-DEP)
1120 CRIT$(PT)=CRIT$
1130 RETURN
1140 '----------------------------------------
```

LISTE DE CABLAGE

Problème : il s'agit de relier entre eux, différents points d'une carte à wrapper :

- Chaque groupe de points est défini par un nom (équipotentielle)
- Les points de wrapping doivent être reliés en quinconce.

On entre d'abord tous les points à relier en fournissant pour chacun d'eux :

- Le nom de l'équipotentielle à laquelle il appartient
- Ses coordonnées d'implantation (abscisse, ordonnée)

Un tri est ensuite effectué sur la table des équipotentielles de façon à rapprocher tous les points appartenant à la même equipotentielle.

L'édition des points est faite d'abord pour la couche basse puis pour la couche haute.

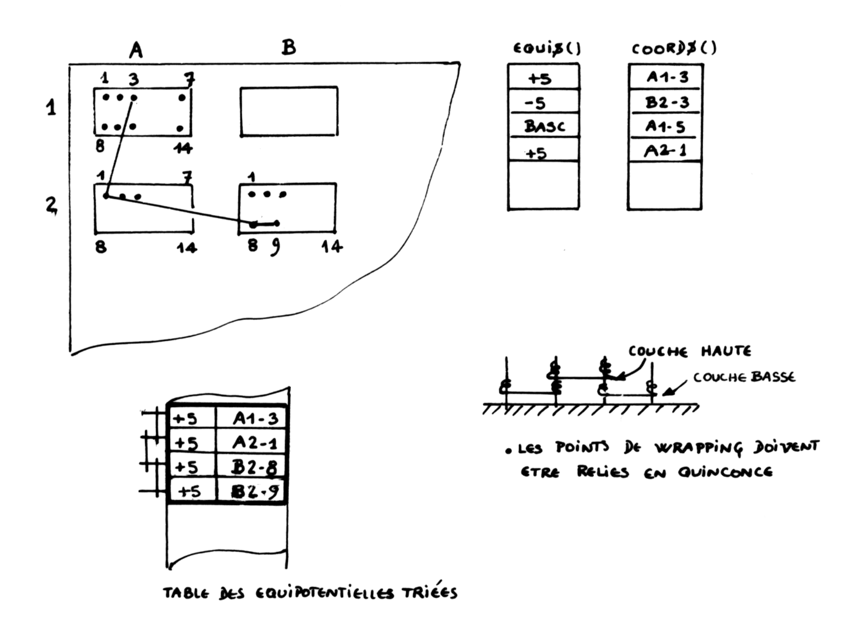


TABLE DES EQUIPOTENTIELLES TRIEES

```
20 '           Programme de LISTE de CABLAGE
30 '           ============================
40 '
45 NOMBRE=8                               ' 8 points a cabler
50 DIM EQUI$(NOMBRE+2),COORD$(NOMBRE+2)
54 '------------------------- Points a cabler
60 DATA +5,A1-3,+5,A2-1
62 DATA +5,B2-8,-5,B2-3
63 DATA -5,C3-2,ALPHA,B3-5
64 DATA BASC,A1-5,BASC,A4-3
69 '
70 FOR I=1 TO NOMBRE
80   READ EQUI$(I)
90   READ COORD$(I)
100 NEXT I
130 ' ----------------------------------------------- Tri des tables
140 K=NOMBRE-1
150 INVERSION=0
160     FOR I=1 TO K
170     IF EQUI$(I+1)+COORD$(I+1)<EQUI$(I)+COORD$(I)
               THEN SWAP EQUI$(I+1),EQUI$(I):SWAP COORD$(I+1),COORD$(I):INVERSION=1
180     NEXT I
190 IF INVERSION=1 THEN K=K-1: GOTO 150
200 '
210 LPRINT:FOR I=1 TO NOMBRE:LPRINT EQUI$(I),COORD$(I):NEXT I:LPRINT
220 ' -----------------------------------------------------------------------------
230 X=1
250 LPRINT "Liste de CABLAGE"
260 LPRINT "================":LPRINT
270 '
280 IF X>=NOMBRE THEN STOP
290 DEBEQUI=X             ' DEBEQUI:memoire de X
300 '------------- Bas
310 IF EQUI$(X+1)=EQUI$(X) THEN LPRINT EQUI$(X),COORD$(X);" ";COORD$(X+1);" Bas"
320 IF EQUI$(X+1)=EQUI$(X) AND EQUI$(X+2)=EQUI$(X+1) THEN X=X+2:GOTO 310
330 '------------- Haut
340 X=DEBEQUI+1
350 IF EQUI$(X+1)=EQUI$(X) THEN LPRINT EQUI$(X),COORD$(X);" ";COORD$(X+1);" Haut"
360 IF EQUI$(X+1)=EQUI$(X) AND EQUI$(X+2)=EQUI$(X+1) THEN X=X+2:GOTO 350
370 IF EQUI$(X+1)><EQUI$(X) AND EQUI$(X+2)=EQUI$(X+1) THEN X=X+1:LPRINT :GOTO 280 E
LSE X=X+2:LPRINT:GOTO 280
380 '
390 '
```

```
+5             A1-3
+5             A2-1
+5             B2-8
-5             B2-3
-5             C3-2
ALPHA          B3-5
BASC           A1-5
BASC           A4-3

Liste de CABLAGE
================

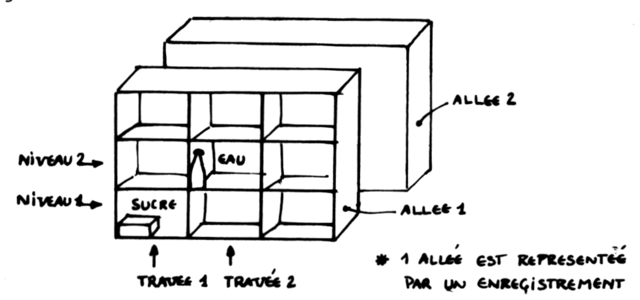
+5             A1-3 A2-1 Bas
+5             A2-1 B2-8 Haut

-5             B2-3 C3-2 Bas

BASC           A1-5 A4-3 Bas
```

GESTION DE CASIERS DE RANGEMENT

Cet exemple de gestion de casiers de rangement devrait nous permettre de mettre en évidence la puissance de l'instruction FIELD# qui, en autorisant la définition de tables à plusieurs dimensions sur la mémoire tampon des fichiers, simplifie la programmation.



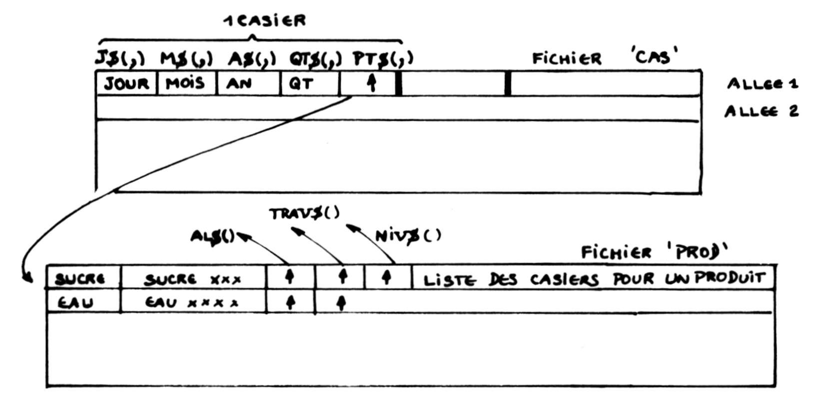
Nous représentons chaque allée par un enregistrement de 256 caractères. Les informations concernant chaque casier qui doivent être stockées sont la date et la quantité.

Chacune de ces informations est stockée dans une table à deux dimensions : J$(TRAVEE,NIV) - M$(TRAVEE,NIV) - etc... Les informations relatives à chaque produit telles que son code et son libellé sont stockées dans un fichier 'PROD'.

Les 'liens' entre les deux fichiers 'CAS' et 'PROD' permettent de retrouver :

1/ pour un casier, le code et le libellé associés (pointeurs PT$(,))

2/ pour chaque produit, tous les casiers où il existe (pointeurs AL$(),TRAV$(),NIV$())



```
10 '
20 '
30 '                    RANGEMENT
40 '                    =========
50 '
60 '    .Permet de gerer des casiers de rangement
70 '    .Chaque allee est representee par un enregistrement
80 '
90 '
100 '
110 CLEAR(2000)
120 DIM AL$(20),TRAVEE$(20),NIV$(20)
130 '
140 '
150 OPEN "R",1,"CAS"                        ' Fichier des casiers de rangement
160 FIELD #1,127 AS D$,1 AS TEM$            ' 253 AU LIEU DE 127 SUR TRS80
170 OPEN "R",2,"PROD"                       ' Fichier des produits
180 FIELD #2,12 AS PP$,1 AS CLAS$,30 AS LIB$
190 FOR I=1 TO 20                           ' Pointeurs vers stock
200   FIELD #2,60 AS D$,3*(I-1) AS D$,1 AS AL$(I),1 AS TRAVEE$(I),1 AS NIV$(I)
210 NEXT I
220 '------------------------------------------------------------------------------
230 '         TEM$ : Temoin d'initialisation
240 GET #1,1:IF TEM$="*" GOTO 500
250 PRINT:PRINT "INITIALISATION DU FICHIER 'CAS'":PRINT
260 FOR I=1 TO 20:GET #1,LOF(1)+1:PUT #1,I:NEXT I   ' Initialisation fichier avec 0
 ASCII
270 LSET TEM$="*":PUT #1,1
280 '------------------------------------------------------------------------------
290 '
300 '   ALLEE    : ALLEE
310 '   TRAVEE   : TRAVEE
320 '   NIV      : NIVEAU
330 '
340 '  FICHIER 'CAS':
350 '
360 '     J$(TRAVEE,NIV)   : JOUR
370 '     M$(TRAVEE,NIV)   : MOIS
380 '     A$(TRAVEE,NIV)   : ANNEE
390 '     QT$(TRAVEE,NIV)  : QUANTITE
400 '     PT$(TRAVEE,NIV)  : POINTEURS VERS PRODUIT
410 '
420 '  FICHIER 'PROD':
430 '
440 '     PP$         : CODE PRODUIT
450 '     AL$(X)      : POINTEURS VERS ALLEE   (TABLE DIM(20))
460 '     TRAVEE$(X)  : POINTEURS VERS TRAVEE
470 '     NIV$(X)     : POINTEURS VERS NIVEAU  (TABLE DIM(20))
480 '
490 '
500 FOR TRAVEE=1 TO 5                 ' 10 SUR TRS80
510   FOR NIV=1 TO 3
520      FIELD #1,24*(TRAVEE-1) AS D$,8*(NIV-1) AS D$,1 AS J$(TRAVEE,NIV),1 AS M$(
TRAVEE,NIV),1 AS A$(TRAVEE,NIV),2 AS QT$(TRAVEE,NIV),1 AS PT$(TRAVEE,NIV)
530   NEXT NIV
540 NEXT TRAVEE
550 '
560 '------------------------------------------------------------------------------
570 '
580 '
590 '
600 '
```

```
650 '
660 '                                        MENU
670 '
680 PRINT TAB(20) "ST  :  STOCKAGE"
690 PRINT
700 PRINT TAB(10):INPUT "MODE? ";M$
710 IF M$="ST" THEN GOSUB 780     ' Stockage
720 IF M$="V" THEN GOSUB 1290     ' Visualisation stockage
730 IF M$="VP" THEN GOSUB 1450    ' Visualisation produit
740 GOTO 680
750 '
760 '=========================================================================
770 '                                                            STOCKAGE
780 PRINT:INPUT "PRODUIT? ";P$
790 IF P$="" THEN RETURN               ' Fin de mode
800 FOR PD=1 TO LOF(2)                 ' Recherche produit
810   GET #2,PD
820   IF P$=LEFT$(PP$,LEN(P$)) THEN GOTO 940
830 NEXT PD
840 '------------
850 PRINT:INPUT "NOUVEAU PRODUIT OK ? ";R$:IF R$><"O" THEN 780
860 GET #2,LOF(2)+1
870 LSET PP$=P$
880 INPUT "CLASSE? ";X$:LSET CL$=X$
890 INPUT "LIBELLE? ";X$:LSET LIB$=X$
900 PD=LOF(2)                          ' LOF(2)+1 SUR TRS80
910 PUT #2,PD
920 '
930 '-------------------------------- Recherche d'un casier libre
940 PRINT:PRINT PP$,LIB$,CL$
950 FOR ALLEE=1 TO 10
960   GET #1,ALLEE                     ' Lecture d'une allee
970   FOR TRAVEE=1 TO 5                ' 10 SUR TRS80
980     FOR NIV=1 TO 3
990       IF CVI(QT$(TRAVEE,NIV))=0 THEN 1060
1000      NEXT NIV
1010    NEXT TRAVEE
1020 NEXT ALLEE
1030 '
1040 PRINT "C'EST PLEIN":STOP
1050 '
1060 PRINT:PRINT "ALLEE;TRAVEE;NIVEAU:";ALLEE;TRAVEE;NIV;"LIBRE ";:INPUT "OK? ";R$:
IF R$<>"O" THEN 780
1070 PRINT:INPUT "QUANTITE? ";QT:LSET QT$(TRAVEE,NIV)=MKI$( QT)
1080 '
1090 INPUT "JOUR? ";X:LSET J$(TRAVEE,NIV)=CHR$(X)
1100 INPUT "MOIS? ";X:LSET M$(TRAVEE,NIV)=CHR$(X)
1110 LSET PT$(TRAVEE,NIV)=CHR$(PD)                          ' pointeur produit
1120 FOR K=1 TO 20                                          ' pointeur casier
1130    IF ASC(AL$(K))=0 THEN LSET AL$(K)=CHR$(ALLEE):LSET TRAVEE$(K)=CHR$(TRAVEE):
LSET NIV$(K)=CHR$(NIV):GOTO 1170
1140 NEXT K
1150 PRINT "PLUS DE PLACE POUR LES POINTEURS":STOP
1160 '
1170 PUT #1,ALLEE
1180 PUT #2,PD
1190 GOTO 780
```

```
1270 '=====================================================================
1280 '                                               VISUALISATION RANGEMENT
1290 LPRINT:LPRINT "OCCUPATION DES CASIERS DE RANGEMENT":LPRINT
1300 FOR ALLEE=1 TO 10
1310   GET #1,ALLEE
1320   LPRINT "Allee:";ALLEE:LPRINT
1330   FOR NIV=1 TO 3
1340     FOR TRAVEE=1 TO 5     ' 10 SUR TRS80
1350       LPRINT USING "###";CVI(QT$(TRAVEE,NIV));
1360       X=ASC(PT$(TRAVEE,NIV)):IF X><0 THEN GET #2,X:LPRINT " ";PP$; ELSE LPRINT SPC(12+1);
1370     NEXT TRAVEE
1380     LPRINT
1390   NEXT NIV
1400   LPRINT
1410 NEXT ALLEE
1420 RETURN
1430 '=====================================================================
1440 '                              VISUALISATION   STOCKAGES POUR UN PRODUIT
1450 PRINT:PRINT:INPUT "QUEL PRODUIT? ";P$
1460 IF P$="" THEN RETURN
1470 FOR PD=1 TO LOF(2)              ' Recherche produit
1480   GET #2,PD
1490   IF P$=LEFT$(PP$,LEN(P$)) THEN 1540
1500 NEXT PD
1510 '
1520 PRINT:PRINT "N'EXISTE PAS":PRINT:GOTO 1450
1530 '
1540 PRINT:PRINT PP$;                ' PP$:code produit
1550 FOR I=1 TO 20                   ' 20 pointeurs pour un produit
1560    IF ASC(AL$(I))=0 THEN 1610
1570    ALLEE=ASC(AL$(I)):GET #1,ASC(AL$(I))         ' Lecture d'une allee
1580    TRAVEE=ASC(TRAVEE$(I))
1590    NIV=ASC(NIV$(I))
1600    PRINT ALLEE;TRAVEE;NIV;"QT:";CVI(QT$(TRAVEE,NIV));"* ";
1610 NEXT I
1620 GOTO 1450
1630 '=====================================================================
```

```
OCCUPATION DES CASIERS DE RANGEMENT

Allee: 1

  5 SUCRE       4 EAU         0             0             0
  6 EAU         0             0             0             0
  8 SUCRE       0             0             0             0

Allee: 2

  0             0             0             0             0
  0             0             0             0             0
  0             0             0             0             0

Allee: 3

  0             0             0             0             0
  0             0             0             0             0
  0             0             0             0             0

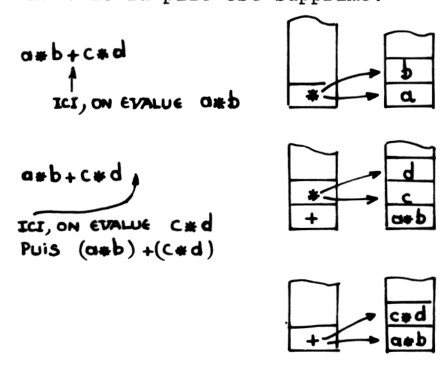
Allee: 4
```

EVALUATION D'EXPRESSIONS

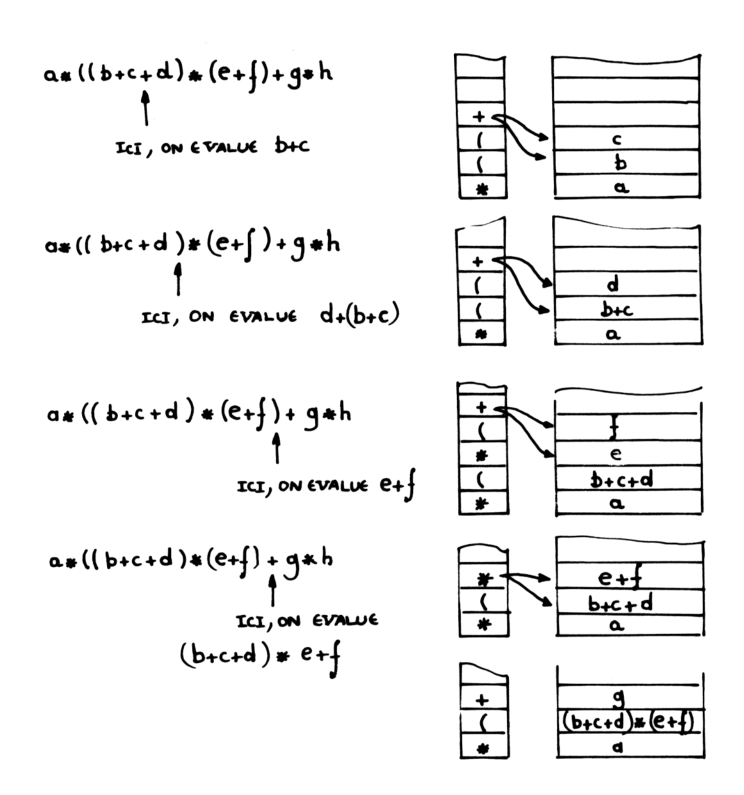
Etudions comment peut être réalisée l'évaluation d'une expression arithmétique. Ce n'est pas, bien sûr, pour le seul plaisir de réinventer ce que fait un compilateur ou un interpréteur. Nous aurons en effet besoin plus loin, d'évaluer des expressions de 'listes' (cf. langage d'interrogation de fichiers).

L'analyse d'une expression se fait de gauche à droite. Chaque opérande rencontré est mis dans une pile dite 'pile des opérandes', chaque opérateur étant lui stocké dans une 'pile des opérateurs'.

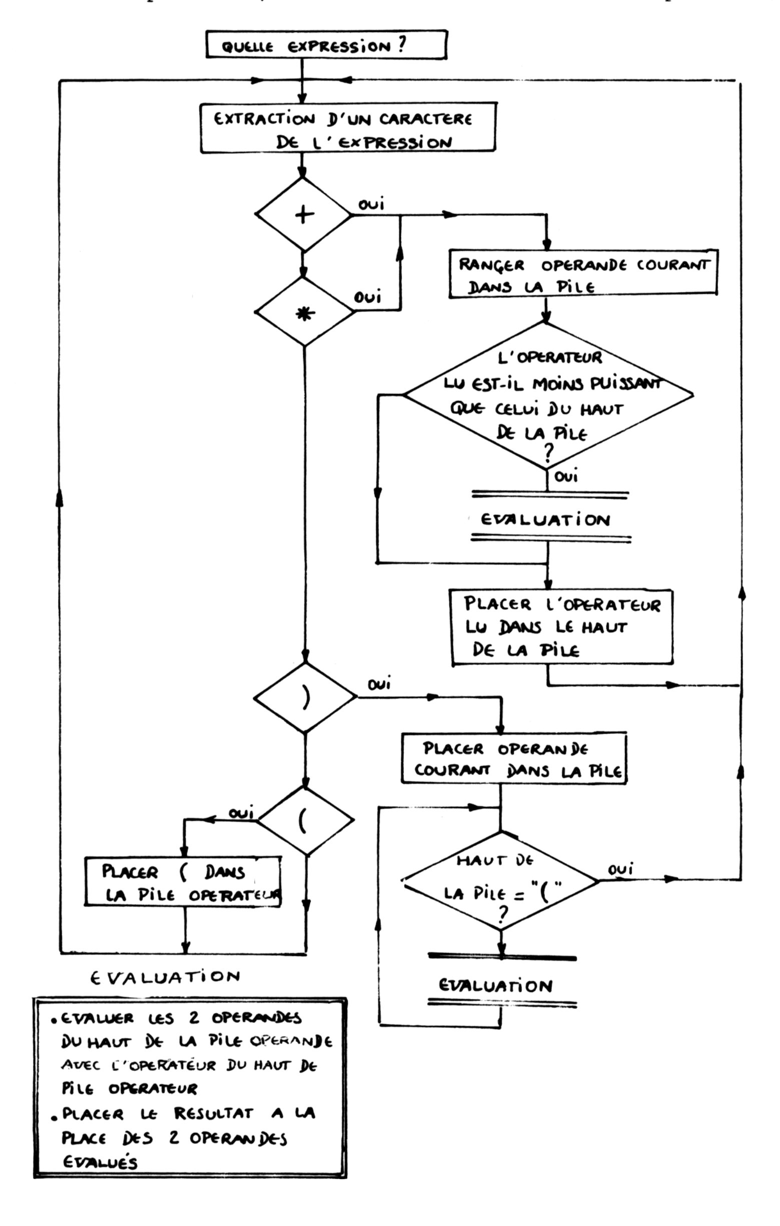
A chaque fois qu'un opérateur de puissance inférieure ou égale à celle de l'opérateur du haut de la pile est rencontré dans l'expression à évaluer, (+ est moins puissant que *), on évalue le résultat de l'opération entre les deux opérandes du haut de la pile avec l'opérateur du haut de la pile, puis le résultat est rangé à la place des deux précédents opérandes. Enfin, l'opérateur du haut de la pile est supprimé.



Lorsque l'expression à évaluer comporte des parenthèses, celles ouvrantes sont stockées systématiquement dans la pile des opérateurs. Considérées plus puissantes que les opérateurs, elles empêchent l'évaluation tant qu'une parenthèse fermante n'est pas rencontrée. Chaque parenthèse fermante provoque l'évaluation jusqu'à la parenthèse ouvrante correspondante.



Remarques sur le programme : compte tenu des temps de traitement des chaînes de caractères, ce programme est relativement long à exécuter. Nous n'avons considéré que les opérations d'addition et de multiplication, celles dont nous aurons besoin plus loin.

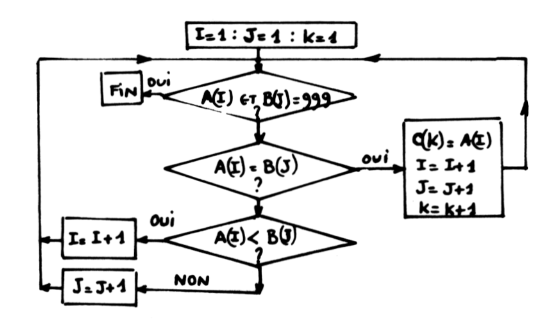


```
30 ' EVA 12.5.80                         EVALUATION D'UNE EXPRESSION
40 '                                     ---------------------------
50 '
60 '      RTEUR$: pile operateurs      RT : pointeur pile operateurs
70 '      RANDE : pile operande        RD : pointeur pile operandes
80 '
90 X$="2*((1+2+3)*(2+3))+15"                     ' Expression a evaluer
100 DEF FNY(Y$)=INSTR("  *+)(",Y$)               ' Affectation d'un poids a *,+,),(
105 '                                                       (1,2,3,4)
110 '
120 RT=1:RD=1       ' RT:pointeur pile operateur/ RD:pointeur pile operande
130 CUR=1           ' Curseur
140 '
150 DEP=CUR         ' Memoire ancienne position curseur
160 '
170 Y$=MID$(X$,CUR,1)        ' Y$: caractere analyse
180 '
190 IF Y$="" THEN Y$=")": GOSUB 280:LPRINT:LPRINT "RANDE(1)=";RA(1):STOP
200 IF Y$="+" THEN GOSUB 280:CUR=CUR+1:GOTO 150
210 IF Y$="*" THEN GOSUB 280:CUR=CUR+1: GOTO 150
220 IF Y$=")" THEN GOSUB 330:RT=RT-1::CUR=CUR+1::GOTO 150
230 IF Y$="(" THEN RTEUR$(RT)=Y$:RT=RT+1:CUR=CUR+1:GOTO 150
240 CUR=CUR+1
250 GOTO 170
260 '---------------------------------------------------------------------
270 '                                                          PLUS/MULT
280 GOSUB 440                                         ' Appel pile operande
290 IF RT>1 AND FNY(Y$)>=FNY(RTEUR$(RT-1))  THEN GOSUB 380:RT=RT-1 :GOTO 290
300 RTEUR$(RT)=Y$:RT=RT+1
310 RETURN
320 '---------------------------------------------------- Parenthese fermante
330 GOSUB 440
340 IF RT>1 AND FNY(Y$)>=FNY(RTEUR$(RT-1)) THEN GOSUB 380:RT=RT-1:GOTO 340
350 RETURN
360 '----------------------------------------------------------- Evaluation
380 FOR I=1 TO RT-1:LPRINT RTEUR$(I);:NEXT I
390 LPRINT TAB(10); :FOR I=1 TO RD-1:LPRINT RANDE(I);:NEXT I:LPRINT TAB(30);RT;RD
400 IF RTEUR$(RT-1)="*" THEN RANDE(RD-2)=RANDE(RD-1)*RANDE(RD-2):RD=RD-1:RETURN
410 IF RTEUR$(RT-1)="+" THEN RANDE(RD-2)=RANDE(RD-1)+RANDE(RD-2):RD=RD-1:RETURN
420 '-------------------------------------------------- Ajout pile operande
440 CH$=MID$(X$,DEP,CUR-DEP)      ' CH$:operande
450 IF CH$="" THEN RETURN
460 RANDE(RD)=VAL(CH$)
470 RD=RD+1
480 RETURN
490 '                   ! + !    ! 3 !
500 '                   ! ( !    ! 2 !    <-- RD
510 '          RT -->   ! ( !    ! 1 !
520 '                   ! * !    ! 2 !
530 '                   -----    -----
540 '                 operateurs   operandes
550 '                  RTEUR$()     RANDE()

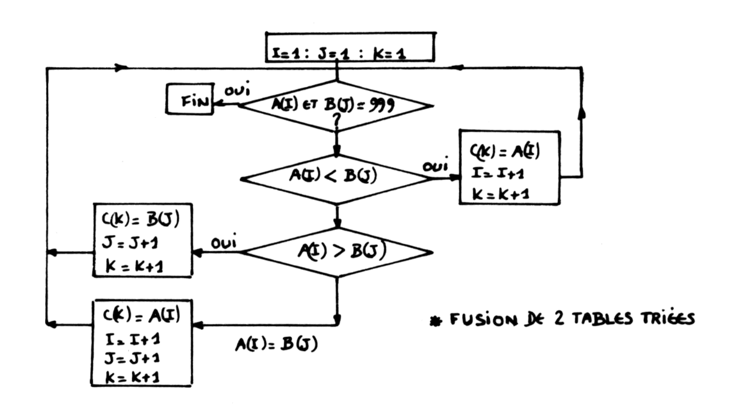
*((+        2  1  2              5  4
*((+        2  3  3              5  4
*(*(+       2  6  2  3           6  5
*(*         2  6  5              4  4
*           2  30                2  3
+           60  15               2  3

RANDE(1)= 75
```

```
10 ' INTER  INTERSECTION DE 2 TABLES TRIEES
20 '        ==============================
30 '
40 '
50 '     Principe: On fait progresser J tant que A(I)>B(J) et I tant que A(I)<B(J)
60 '    Des 99999 ont ete places a la fin des tables A() et B() afin de simplifier
70 '    le traitement
80 '
90 '         TABLE A     TABLE B     ----->   TABLE C
100 '        ---------   ----------           ---------
110 '        |  2     |   |   1      |          |   2    |
120 '        |  4     |   |   2      |          |   4    |
130 '     I->|  6     | J->|  3      |          |        |
140 '        |  8     |   |   4      |          |        |
150 '        |        |   |          |          |        |
160 '        | 9999   |   |          |          |        |
170 '                     | 9999     |          |        |
180 '                                           |        |
190 '
200 DIM A(20),B(20),C(40)
210 '
220 FOR I=1 TO 10 :A(I)=2*I:NEXT I              ' Sequence d'essai
230 FOR I=1 TO 15:B(I)=I:NEXT I
240 FOR I=1 TO 20:PRINT A(I),B(I):NEXT I
250 '
260 A(11)=9999:B(16)=9999:GOSUB 300:FOR I=1 TO K-1:PRINT C(I):NEXT I
270 STOP
280 '---------------------------------------------------------------------
290 '                                INTERSECTION DE A() ET B()
300 I=1:J=1:K=1
310 '
320 IF A(I)=9999 AND B(J)=9999 THEN RETURN
330 '
340 IF A(I)=B(J) THEN C(K)=A(I):I=I+1:J=J+1:K=K+1:GOTO 320  ELSE
      IF A(I)<B (J) THEN I=I+1:GOTO 320 ELSE
         IF A(I)>B(J) THEN J=J+1:GOTO 320
```



```
10 ' FUSI       FUSION de 2 TABLES TRIEES
20 '            ========================
30 '
40 '   Principe: On analyse les tables A et B en faisant progresser
50 '             I si A(I)<B(J)  et J si A(I)>B(J)
60 '
70 '   De facon a simplifier le traitement,nous placons des 9999
80 '   a la fin des tables A() et B()
85 '   Pour trier des chaines de caracteres,nous placerions des 'ZZZZZZ'
90 '
100 '
110 '          TABLE  A     TABLE  B     ---->   TABLE  C
120 '          ---------    ----------           ---------
130 '          ¦  2     ¦   ¦   1     ¦          ¦   1    ¦
140 ' I -->    ¦  4     ¦   ¦   2     ¦          ¦   2    ¦
150 '          ¦  6     ¦   ¦   3     ¦          ¦   3    ¦
160 '          ¦  8     ¦J-->¦  4     ¦    K-->! 4     ¦
170 '          ¦        ¦   ¦         ¦          ¦        ¦
180 '          ¦ 9999   ¦   ¦         ¦          ¦        ¦
190 '                       ¦ 9999    ¦          ¦        ¦
200 '                                            ¦        ¦
210 '
220 DIM A(20),B(20),C(40)
230 '
240 FOR I=1 TO 10 :A(I)=2*I:NEXT I              ' Sequence d'essai
250 FOR I=1 TO 15:B(I)=I:NEXT I
260 FOR I=1 TO 20:PRINT A(I),B(I):NEXT I
270 '
280 A(11)=9999:B(16)=9999:GOSUB 320:FOR I=1 TO K-1:PRINT C(I):NEXT I
290 STOP
300 '---------------------------------------------------------------------
310 '                                             FUSION DE A() et B()
320 I=1:J=1:K=1
330 '
340 IF A(I)=9999 AND B(J)=9999 THEN RETURN
350 '
360 IF A(I)<B(J) THEN C(K)=A(I):I=I+1:K=K+1:GOTO 340  ELSE
     IF B(J)<A(I) THEN C(K)=B(J):K=K+1:J=J+1:GOTO 340 ELSE
       IF A(I)=B(J) THEN C(K)=A(I):K=K+1:I=I+1:J=J+1:GOTO 340
```

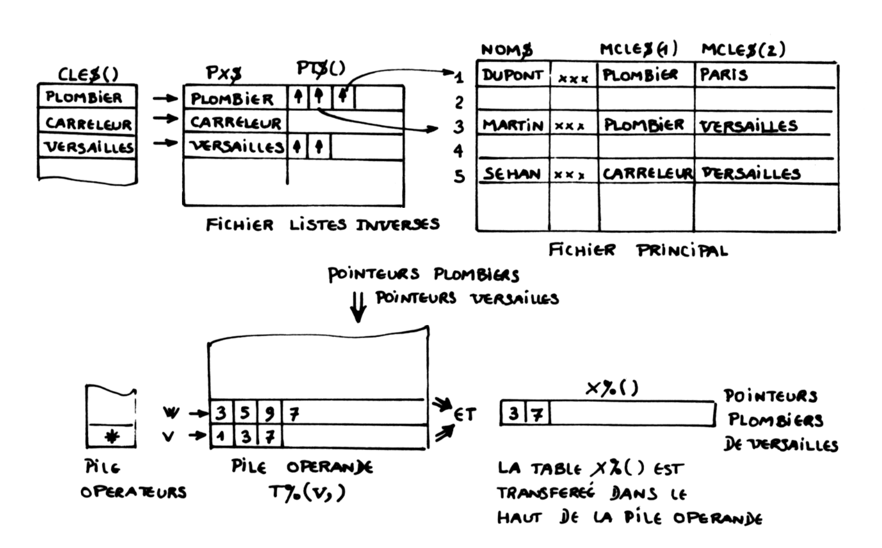


INTERROGATION DE FICHIER PAR UNE EXPRESSION

(cf. listes inverses et évaluation d'une expression)

Nous vous proposons de réaliser ici, aussi simplement que possible, l'interrogation d'un fichier par une expression.

Contrairement à l'exercice précédent (INTF) où nous analysions systématiquement chaque enregistrement du fichier pour déterminer s'il répondait ou non aux critères recherchés, nous utilisons des *listes inverses* précédemment constituées à l'ajout de chaque enregistrement) qui pointent vers les seuls enregistrements susceptibles de répondre aux critères demandés.



Pour simplifier la programmation, nous avons banalisé les mots-clés des professions et des villes, sachant qu'ils ne sauraient coincider. (Il y a peu de chance pour qu'un nom de profession coincide avec un nom de ville).

RECHERCHE DU TYPE 'ET'

Exemple : Pour retrouver tous les enregistrements correspondant aux *PLOMBIERS* de *VERSAILLES,* nous recherchons tous les pointeurs des *plombiers* ainsi que ceux des personnes habitant
Nous les rangeons dans une table T%(,), leur intersection fournit la liste des *PLOMBIERS* de *VERSAILLES.*

Jusqu'à 110 pointeurs par mot-clé sont disponibles avec le programme présenté.

EVALUATION D'UNE EXPRESSION

Un mode plus élaboré permet d'évaluer une expression du type : PLOMB*(VERS+BOUL)

PLUSIEURS MOTS-CLES PAR ENREGISTREMENT DU FICHIER LISTES INVERSES

Pour économiser l'espace disque occupé par le fichier listes inverses, plusieurs mots-clés par enregistrement sont définis (3 par exemple). La table des pointeurs PT$() à une dimension devient une table PT$(,) à deux dimensions.

```
PX$(1)      PT$(1,x)  PX$(2)  PT$(2,x)          PT$(3,x)
|PLOMBIER|↑|↑| | |EPICIER|↑|↑| | |VERSAILLES| | | | |
```

.L'instruction FIELD # s'ecrit:

```
FOR J=1 to 3
  FOR I=1 to 30              ' 30 pointeurs par mot-cle
    FIELD #2,84*(J-1) AS D$,15 AS PX$(J),2*(I-1) AS D$,2 ASD PT$(J,I)
  NEXT I
NEXT J
```

a/ .La constitution des tables CLE$() et IX%() en debut de session devient:

```
910   PRINT "MOT-CLE"
920   FOR NI=1 to LOF(2)
      GET #1,NI
      FOR P=1 TO 3
        IF ASC(PX$(P))=0 THEN GOTO 950
        PRINT PX$(P):NB=NB+1:CLE$(NB)=PX$(P):IX%(NB)=NI*3+P
      NEXT P
950   NEXT NI
```

.On code dans IX%() la valeur NI*3+P

```
IX%()
|       |
|NI*3+P |
|       |
```

b/ La mise a jour des listes inverses se fait par:

```
CLE$()    IX%()          P=1             P=2              P=3
|EPICIER| |NI*3+p| NI--> |PLOMBIER|↑|↑|EPICIER|↑| | |PARIS |↑|↑| |
```

NI: No enregistrement du fichier listes inverses
P : Position dans cet enregistrement (1,2,3)
NB: Nombre de cles dans la table CLE$()

```
710 FOR V=1 TO NMCLES
720   IF ASC(MCLE$(V))=0 THEN GOTO 860
730   PRINT MCLE$(V)
740   FOR K=1 TO 150                              ' Recherche du mot-cle
750     IF MCLE$(V)=CLE$(K) THEN NI=INT((IX%(K)-1)/3):
          P=((IX%(K)-1) MOD 3)+1:GOTO 810         ' Le mot-cle existe
760     IF CLE$(K)="" THEN P=((K-1) MOD 3)+1:NI=INT((K-1)/3)+1:
          CLE$(K)=MCLE$(V):IX%(K)=NI*3:NB=NB+1:GOTO 810  'Ajout mot-cle
770   NEXT K


810   GET #2,NI
820   FOR I=1 TO 30                               ' Ajout pointeur
830     IF CVI(PT$(P,I))=0 THEN LSET PT$(P,I)=MKI$(NE):
          LSET PX$(P)=CLE$(K):PUT #2,NI:GOTO 860
840   NEXT I

860 NEXT V
```

c/ .Pour la recherche dans le fichier listes inverses,le decodage du contenu de IX%()
se fait par:

```
1430   NI=INT((IX%(I)-1)/3)          ' No d'enregistrement fichier liste inverse
1432   P =((IX%(I)-1) MOD 3)+1       ' Position dans l'enreg (1,2,3)
1434   GET #2,NI


1460  NE=CVI(PT$(P,K))               ' NE: No d'enregistrement fichier principal
```

L'allocation dynamique des enregistrements du fichier principal n'ayant pas été prévue, une réorganisation de celui-ci (par retassage) sera faite dès que les suppressions seront devenues trop nombreuses. Il faudra alors recréer les listes inverses (mode 'CRI').

```
10 CLEAR(3000)
20 PRINT "BASE/BAS 6.1.81":PRINT
30 '
40 '  INTERROGATION D'UN FICHIER PAR UNE EXPRESSION
50 '
60 '        SYNTHESE: EVALUATION D'UNE EXPRESSION+LISTES INVERSES
70 '
80 '  EX: PLOMBIER*(VERSAILLES+SEVRES)+CARRELEUR*VERSAILLES
90 '
100 '   A CHAQUE MOT-CLE EST ASSOCIE UNE LISTE DE POINTEURS VERS
110 '   LES ENREGISTREMENTS DU FICHIER PRINCIPAL OU IL EXISTE
120 '
130 '  ON EVALUE UNE EXPRESSION PAR INTERSECTIONS ET UNIONS SUCCESSIVES
140 '  DE LISTES DE POINTEURS.
150 NMCLES=4                ' Nombre de MOTS CLES par enregistrement
160 COM$(1)="MOT-CLE1":COM$(2)="MOT-CLE2":COM$(3)="MOT-CLE3":COM$(4)="MOT-CLE4"
170 DIM PT$(110),CLE$(50)
180 DIM T%(5,110)                    ' Pile operandes (hauteur=5)9
190 DIM X%(200)                      ' Table des operandes
200 OPEN "R",#1,"BASE"               ' Fichier principal
210 FOR I=1 TO NMCLES:FIELD #1,50 AS NOM$,(I-1)*15 AS D$,
     15 AS MCLE$(I):NEXT I
220 OPEN "R",#2,"IBASE"              ' Fichier listes inverses
230 FOR I=1 TO 110 :FIELD #2,15 AS PX$,(I-1)*2AS D$,2 AS PT$(I):NEXT I
240 GOSUB 900
250 '================================================================
260 '                                          MENU
270 PRINT:PRINT "           PENSER A FAIRE  'FIN '  ":PRINT
280 PRINT :INPUT "MODE? (C,CRI,RCLE,INT,LF,FIN,..)";MODE$
290 IF MODE$="C" THEN GOSUB 410        'Creation
300 IF MODE$="CRI" THEN GOSUB 560      'Creation liste inverse
310 IF MODE$="RCLE" THEN GOSUB 1000    'Recherche pour un MOT-CLE
320 IF MODE$="ET" THEN GOSUB 1660      'Mode ET
330 IF MODE$="OU" THEN GOSUB 1770
340 IF MODE$="INT" THEN GOSUB 2540     'Interrogation par une expression
350 IF MODE$="LF" THEN GOSUB 2310
360 IF MODE$="SU" THEN GOSUB 2380
370 IF MODE$="FIN" THEN CLOSE #1,#2:CMD "S"
380 GOTO 280
390 '=================================================================
400 '                                                CREATION(C)
410 PRINT :X$="":INPUT "NOM ";X$:IF X$=""THEN RETURN
420 NE=LOF(1)+1:GET #1,NE               ' Ajout en fin de fichier
430 LSET NOM$=X$
440 '
450 FOR K=1 TO NMCLES
460   Y$="":PRINT COM$(K);:INPUT Y$
470   IF Y$><"" THEN LSET MCLE$(K)=Y$
480 NEXT K
490 '
500 PRINT:INPUT "OK (O/N) ";R$:IF R$<>"O" THEN 410
510 PUT#1,NE
520 GOSUB 680                         ' Appel MAJ listes inverses
530 GOTO 410
540 '=================================================================
550 '                               CONSTITUTION LISTE INVERSE(CRI)
560 CLOSE#2:KILL "IBASE":OPEN "R",2,"IBASE"
570 FOR I=1 TO 50:CLE$(I)="":NEXT I:NB=0
580 FOR NE=1 TO LOF (1)
590   GOSUB 680                       ' Appel MAJ listes inverses
600 NEXT NE
610 CLOSE #2:OPEN "R",2,"IBASE"
620 RETURN
630 '
```

```
670 '-------------------------------------------------------
675 '                                    MAJ LISTE INVERSE
680 GET #1,NE:IF ASC(NOM$)=0 GOTO 870
690 PRINT:PRINT NOM$
700 '
710 FOR V=1 TO NMCLES
720    IF ASC(MCLE$(V))=0 GOTO 860
730     PRINT MCLE$(V)
740     FOR K=1 TO 50                     ' Recherche MOT-CLE
750        IF MCLE$(V)=CLE$(K) THEN GOTO 810
760        IF CLE$(K)="" THEN CLE$(K)=MCLE$(V):NB=NB+1:GOTO 810
770    NEXT K
780 '
790 PRINT"AUGMENTEZ LA TAILLE DE CLE$":STOP
800 '
810 GET #2,K
820 FOR I=1 TO 110                         ' Ajout d'un pointeur
830    IF CVI(PT$(I))=0 THEN LSET PT$(I)=MKI$(NE):
         LSET PX$=CLE$(K):PUT #2,K:GOTO 860 ' Pointeur libre?
840 NEXT I
850 PRINT"Y'A PLUS ASSEZ DE PLACE POUR LES POINTEURS":STOP
860 NEXT V
870 RETURN
880 '==============================================================
890 '                              CONSTITUTION DE LA TABLE CLE$
900 NB=0                           ' NB:nombre de cles
910 PRINT "MOT-CLE":PRINT
920 FOR I=1 TO LOF (2)
930     GET #2,I:IF ASC(PX$)=0 GOTO 950
940     PRINT PX$:NB=NB+1:CLE$(NB)=PX$
950 NEXT I
960 PRINT
970 RETURN
980 '===============================================================
990 '                              INTERROGATION PAR MOT-CLE(RCLE)
1000 PRINT:X$="":INPUT "MOT-CLE CHERCHE ? ";X$:IF X$="" THEN RETURN
1010 '
1020 FOR I=1 TO NB
1030   IF X$=LEFT$(CL$(I),LEN(X$)) THEN GOTO 1080
1040 NEXT I
1050 '
1060 PRINT "MOT-CLE N'EXISTE PAS":PRINT : GOTO 1000
1070 '
1080 GET #2,I:PRINT
1090 '
1100 FOR K=1 TO 110
1110    NE=CVI(PT$(K)):IF NE=0 THEN GOTO 1140
1120    GET #1,NE
1130    PRINT NOM$
1140 NEXT K
1150 GOTO 1000
1160 '
1170 '
```

```
1320 '----------------------------------------------------------------
1325 '                           RECHERCHE POINTEURS POUR UN CRITERE
1330 '
1340 PRINT :X$="":INPUT "MOT-CLE CHERCHE? ";X$:IF X$="" THEN VLD=2:RETURN
1345 CC$(P)=X$:P=P+1
1350 '----
1360 FOR I=1 TO NB
1370     IF X$=LEFT$(CLE$(I),LEN(X$)) THEN 1420
1380 NEXT I
1390 '
1400 PRINT "MOT-CLE N'EXISTE PAS": PRINT:VLD=2:RETURN:GOTO 1340
1410 '
1420 '
1430 GET #2,I
1440 '
1450 FOR K=1 TO 110
1460    NE=CVI(PT$(K)):IF NE=0 THEN T%(V,K)=9999:GOTO 1500
1470    T%(V,K)=NE
1480 NEXT K
1490 '----------  TRI DE TV%(V,)
1500 INV=0
1510 FOR I=1 TO K-1
1520    IF T%(V,I)>T%(V,I+1) THEN X=T%(V,I):
        T%(V,I)=T%(V,I+1):T%(V,I+1)=X:INV=1
1530 NEXT I
1540 IF INV=0 THEN VLD=1:RETURN ELSE GOTO 1500
1550 '----------------------------------------------------------------
1555 '                               INTERSECTION DE 2 LISTES
1560 I=1:J=1:K=1
1580 '                             Le resultat est range DANS T%(V,)
1590 IF T%(V,I)=9999 THEN T%(V,K)=9999:RETURN
1600 IF T%(V,I)<T%(W,J) THEN I=I+1:GOTO 1590
1610 IF T%(V,I)>T%(W,J) THEN J=J+1:GOTO 1590
1620 IF T%(V,I)=T%(W,J) THEN T%(V,K)=T%(V,I):I=I+1:J=J+1:K=K+1:GOTO 1590
1630 STOP
1640 '=================================================================
1650 '                             RECHERCHE DU TYPE 'ET'
1660 FOR I=1 TO 5:CC$(I)="":NEXT I:P=1
1670 V=1:GOSUB 1340:ON VLD GOTO 1690,1680    ' Appel premier MOT-CLE
1680 RETURN
1690 V=2:GOSUB 1340:ON VLD GOTO 1710,1700    ' Appel second MOT-CLE
1700 GOTO 1660
1710 V=1:W=2:GOSUB 1560                      ' Appel intersection
1720 OP$(P-2)="* "
1730 GOSUB 2170
1740 GOTO 1690                               ' Appel edition
1750 '================================================================
1760 '                             RECHERCHE DU TYPE 'OU'
1770 FOR I=1 TO 5:CC$(I)="":NEXT I:P=1
1780 V=1:GOSUB 1340 :ON VLD GOTO 1800,1790   ' Appel premier MOT-CLE
1790 RETURN
1800 V=2:GOSUB 1340:ON VLD GOTO 1820,1810    ' Appel second MOT-CLE
1810 GOTO 1770
1820 V=1:W=2:GOSUB 2040                      ' Appel fusion
1830 OP$(P-2)="+ "
1840 GOSUB 2170                              ' Appel edition
1850 GOTO 1800
1860 '-----------------------------------------------------
1870 '
1880 '
```

```
2010 '
2020 '
2030 '                              FUSION DE 2 LISTES
2040 I=1:J=1:K=1
2050 '
2060 IF T%(V,I)=9999 AND T%(W,J)=9999 THEN X%(K)=9999 :GOTO 2110
2070 IF T%(V,I)<T%(W,J) THEN X%(K)=T%(V,I):I=I+1:K=K+1:GOTO 2060
2080 IF T%(V,I)>T%(W,J) THEN X%(K)=T%(W,J):J=J+1:K=K+1:GOTO 2060
2090 IF T%(V,I)=T%(W,J) THEN X%(K)=T%(V,I):I=I+1:J=J+1:K=K+1:GOTO 2060
2100 '
2110 FOR I=1 TO 200
2120    IF X%(I)=9999 THEN T%(V,I)=X%(I):GOTO 2150
2130    T%(V,I)=X%(I)
2140 NEXT I
2150 RETURN
2160 '--------------------------------------------------------
2165 '                                       EDITION
2170 PRINT:PRINT
2180 FOR I=1 TO 5
2190     IF CC$(I)><"" AND CC$(I+1)><"" THEN
         PRINT CC$(I);OP$(I);:GOTO 2210
2200     IF CC$(I)><"" THEN PRINT CC$(I);
2210 NEXT I
2220 PRINT:PRINT
2230 FOR K=1 TO 200
2240    NE=T%(1,K):IF NE=9999 OR NE = 0 THEN PRINT :GOTO 2270
2250    GET #1,NE:PRINT NE;LEFT$(NOM$,30);MCLE$(1);MCLE$(2)
2260 NEXT K
2270 RETURN
2280 '
2290 '================================================================
2300 '                                          LISTE DU FICHIER(LF)
2310 PRINT:FOR I=1 TO LOF(1)
2320   GET #1,I:IF ASC(NOM$)=0 GOTO 2340
2330 PRINT I;LEFT$(NOM$,30);MCLE$(1);MCLE$(2)
2340 NEXT I
2350 RETURN
2360 '================================================================
2370 '                                     MODE SUPPRESSION(SU)
2380 PRINT:NE=0:INPUT "QUEL ENREGISTREMENT ";NE
2390 IF NE=0 THEN RETURN
2400 GET #1,NE:PRINT NOM$;:R$="":INPUT " ANNULE OK (O/N) ";R$:
    IF R$<>"O" THEN 2380
2410 GET #1,LOF(1)+1:PUT #1,NE
2420 GOTO 2380
2430 '==============================================================
2440 '
2441 '
2442 '
2444 '
2445 '
2446 '
2447 '
2450 '
2460 '
2465 '
2466 '
2470 '
2480 '
2490 '
2500 '
```

```
2520 '
2525 '
2526 '
2530 '                    INTERROGATION PAR UNE EXPRESSION
2535 '
2540 DEF FNY(Y$)=INSTR(" *+)(",Y$)     ' Affectation d'un poids A *,+
2545 '
2560 PRINT:EX$="":INPUT "EXPRESSION ";EX$
2570 IF EX$="" THEN RETURN
2580 '
2590 RT=1:RD=1                 ' Pointeurs piles operateurs et operandes
2600 CUR=1                     ' Curseur
2610 DEP=CUR
2620 '
2630 Y$=MID$(EX$,CUR,1)        ' Caractere analyse
2640 '
2650 IF Y$="" THEN Y$=")":GOSUB 2730:IF VLD=1 THEN GOSUB 2220:GOTO 2560
     ELSE  GOTO 2560
2660 IF Y$="+" OR Y$="*" THEN GOSUB 2730:CUR=CUR+1:GOTO 2610
2670 IF Y$=")" THEN GOSUB 2780:RT=RT-1:CUR=CUR+1:GOTO 2610
2680 IF Y$="(" THEN RTEUR$(RT)=Y$:RT=RT+1:CUR=CUR+1:GOTO 2610
2690 CUR=CUR+1
2700 GOTO 2630
2710 '---------------------------------- PLUS/MULT
2720 '
2730 GOSUB 2870:IF VLD=2 THEN RETURN     ' Appel pile operande
2740 IF RT>1 AND FNY(Y$)>=FNY(RTEUR$(RT-1)) THEN
      GOSUB 2830:RT=RT-1:GOTO 2740
2750 RTEUR$(RT)=Y$:RT=RT+1
2760 RETURN
2770 '---------------------------------- PARENTHESE FERMANTE
2780 GOSUB 2870                          ' Appel pile operande
2790 IF RT>1 AND FNY(Y$)>=FNY(RTEUR$(RT-1)) THEN
      GOSUB 2830:RT=RT-1:GOTO 2790
2800 RETURN
2810 '---------------------------------- EVALUATION
2820 '
2830 IF RTEUR$(RT-1)="*" THEN V=RD-2:W=RD-1:GOSUB 1560:RD=RD-1:RETURN
2840 IF RTEUR$(RT-1)="+" THEN V=RD-2:W=RD-1:GOSUB 2040:RD=RD-1:RETURN
2850 '-----------------------------------------------------------------
2860 '                          AJOUT PILE OPERANDE
2870 CH$=MID$(EX$,DEP,CUR-DEP)      ' Operande
2880 IF CH$="" THEN RETURN
2890 X$=CH$:V=RD:GOSUB 1360:IF VLD=2 THEN RETURN
2900 RD=RD+1
2910 RETURN
2920 '
2930 '           !   !    !           !
2940 '           !   !    !           !
2950 '   RT ---> !   !    ! POINTEURS!  <-- RD
2960 '           ! * !    ! POINTEURS!
2970 '           -----    ------------
2980 '           rteur$()   t%(,)
2990 '         OPERATEURS   OPERANDES
3000 '=================================================================
```

INDEX A DEUX NIVEAUX

(cf. Basic et ses Fichiers, tome 1, p. 99)

Un index continu sur disque en ordre croissant permet, grâce à une table d'index réduite en mémoire centrale, de retrouver une clé par un seul accès à l'index sur disque. Mais l'insertion d'une nouvelle clé oblige à décaler sur disque toutes les clés de l'index en aval de l'insertion, ce qui peut être très long.

L'index à deux niveaux, par l'insertion dynamique de blocs dans l'index, évite ces décalages. La régénération d'un index en cas de destruction de celui-ci ou d'incohérence avec le fichier principal, doit être prévue.

Deux solutions sont possibles :

- On explore séquentiellement le fichier principal, et pour chaque clé, nous appelons les routines de recherche et de création de clé (déjà écrites). Cette solution simple à mettre en oeuvre conduit à des temps d'exécution longs.
- Toutes les clés du fichier sont amenées en mémoire centrale puis triées. Il suffit alors de sauvegarder cette table dans l'index. Rapide, cette méthode nécessite de la place mémoire pour les tables des clés et d'index.

```
10 ' ID3 25.12.80
20 '
30 '                               INDEX a 2 NIVEAUX
40 '                               =================
50 '
60 '      cf BASIC et ses FICHIERS p 99
70 '
130 '
140 '         CLE$()    INDEX()      FICHIER INDEX        FICHIER PRINCIPAL
150 '        -------   --------    ----------------      -------------------
160 '        |DUPO |->|   3   |   1| Table CLE$    |    |LUCET  |xxxxxxxxxxx|
170 ' LIGNE->|MART |->|   2   |   2|MART|^|MORI|^|  -->|DUPOND |xxxxxxxxxxx|
180 '        |ORSI |  |       |   3|DUPO|^|DUPO|^|    |MARTIN |XXXXXXXXXXX|
185 '        |  *  |  |       |   4|    | |    | |    |       |           |
190 '
200 '                              SCLE$()  PT$()         NM$
210 '
220 '
225 '            SCLE$(): Table definie dans FIELD# du fichier index
230 '            RANG   : Adresse de rangement dans le fichier principal
240 '            ISER   : Position d'insertion dans un enregistrement de l'index
245 '
250 '   Au depart ,la table CLE$() est initialisee avec des '*'
260 '
270 ' Seules les premieres lettres des CLES sont enregistrees dans le fichier index
280 '  et la table  CLE$() est sauvegardee dans l'enreg no 1 du fihier index
290 '
310 CLEAR(5000)
320 DIM TC$(100),IX%(100)
325 LGCL=5                    ' LGCL:Longueur des cles dans le fichier index
330 NCL=10                    ' NCL :(254/(LGCL+2) SUR TRS80 (Nb de cles par enreg)
350 DIM PT$(NCL),SCLE$(NCL)
360 '
370 DIM CLE$(NCL),INDEX(NCL)
380 OPEN "R",1,"TOT"          ' Fichier principal
390 OPEN "R",2,"ITO"          ' Fichier index
400 FIELD #1,10 AS NM$,8 AS PREN$,20 AS TEL$
410 FOR I=1 TO NCL :FIELD #2,(LGCL+2)*(I-1) AS D$,(LGCL) AS SCLE$(I),2 AS PT$(I):NE
XT I
420 GOSUB 790
430 '
440 '
450 PRINT:INPUT "Mode? (C,RI,LISTT,SUP,CRI..) ";M$
460 IF M$="C" THEN GOSUB 550
480 IF M$="LISTT" THEN GOSUB  1700
490 IF M$="SUP" THEN GOSUB 1820
500 IF M$="CRI" THEN GOSUB 2010       ' Creation index (si incident)
510 GOTO 450
530 '=============================================================================
550 PRINT:INPUT "Nom? ";NOM$:IF NOM$="" THEN RETURN           ' AJOUT d'UN NOM
555 GOSUB 960:ON R GOTO 560,570
560 PRINT:PRINT NM$:PRINT:GOTO 550                    ' Existe deja
565 '
570 RANG=LOF(1):GET #1,RANG                           ' LOF(1)+1 SUR TRS80
580 LSET NM$=NOM$
600 INPUT "Prenom?    ";X$:LSET PREN$=X$
630 INPUT "Telephone? ";X$:LSET TEL$=X$
650 '
660 PRINT:PRINT TAB(30):INPUT "OK? ";R$:IF R$><"O" GOTO 550
670 PUT #1,RANG
690 GOSUB 1340                                  ' Appel MAJ index
700 GOTO 550
710 '
```

```
720 '=========================================================================
790 '             Lecture de CLE$()  sauvegarde dans enreg 1 de l'index
800 GET#2,1
810 IF ASC(SCLE$(1))=0 THEN PUT #2,1:PRINT "INITIALISATION INDEX"
820  FOR I=1 TO NCL
830   IF ASC(SCLE$(I))><0 THEN CLE$(I)=SCLE$(I):INDEX(I)=CVI(PT$(I)) ELSE CLE$(I)="
*"
840  NEXT I
850 RETURN
860 '
870 '-------------------------------------------------------------------
880 '                          Sauvegarde de CLE$()
890    FOR I=1 TO NCL:LSET SCLE$(I)=CLE$(I):LSET PT$(I)=MKI$(INDEX(I)):NEXT I
900    PUT #2,1:RETURN
910 RETURN
920 '=========================================================================
930 '                                                             RECHERCHE CLE
940 '
950 '
960 L=LEN(NOM$)
970 '
980 IF LEFT$( CLE$(1),1)="*" THEN ALC=1:DECAL=0:LIGNE=1:R=2:RETURN
990 FOR I=1 TO NCL
1000     IF NOM$=LEFT$(CLE$(I),L) OR CLE$(I)=LEFT$(NOM$,LGCL) THEN LIGNE=I-1:GOTO 10
80
1010     IF NOM$<CLE$(I) THEN LIGNE=I-1:GOTO 1080
1020     IF LEFT$(CLE$(I),1)="*" THEN LIGNE=I-1:GOTO 1080
1030 NEXT I
1040 STOP
1050 '              Retour:   R=1 :la cle existe / R=2 : elle n'existe pas
1070 '
1080 IF LIGNE<1 THEN LIGNE=1
1090 GET #2,INDEX(LIGNE)
1100 '
1110 FOR J=1 TO NCL
1120   IF NOM$<LEFT$(SCLE$(J),L) THEN ISER=J:R=2:GOTO 1220
1130   IF CVI(PT$(J))=0 THEN IF NOM$<LEFT$(CLE$(LIGNE+1),L) OR LEFT$(CLE$(LIGNE+1),
1)="*" THEN  R=2:ISER=J:ALC=0:DECAL=0:RETURN ELSE LIGNE=LIGNE+1:GOTO 1090
1140   IF NOM$=LEFT$(SCLE$(J),L) OR SCLE$(J)=LEFT$(NOM$,LEN(SCLE$(J)))  THEN GOTO 1
150 ELSE GOTO 1180
1150   GET #1,CVI(PT$(J)):IF NOM$=LEFT$(NM$,L) THEN RANG=CVI(PT$(J)):ISER=J:R=1:RET
URN
1160 '
1170   IF NOM$<LEFT$(NM$,L) THEN ISER=J:R=2:GOTO 1220
1180 NEXT J
1190 '
1200 R=2:ISER=1:LIGNE=LIGNE+1:IF LEFT$(CLE$(LIGNE),1)="*" THEN ALC=1:DECAL=0:RETURN
 ELSE GET #2,INDEX(LIGNE):GOTO 1090
1210 '
1220 IF CVI(PT$(NCL))=0 THEN ALC=0:DECAL=0:R=2:RETURN
1230 IF LEFT$(CLE$(LIGNE+1),1)="*" THEN ALC=1:DECAL=1:R=2:RETURN
1240 GET #2,INDEX(LIGNE+1)
1250 IF CVI(PT$(NCL))><0 THEN ALC=1:DECAL=1:R=2:RETURN
1260 DECAL=1:ALC=0:R=2:RETURN
1320 '=========================================================================
```

```
1329 '
1330 '                                                             AJOUT CLE
1340 IF ALC=1 THEN IX=LOF(2)              ' LOF(2)+1 SUR TRS80
1350 IF ALC=0 AND DECAL=0 GOTO 1400       ' IX:adresse fichier index
1360 IF ALC=1 AND DECAL=0 GOTO 1460
1370 IF ALC=0 AND DECAL=1 GOTO 1580
1380 IF ALC=1 AND DECAL=1 GOTO 1530
1390 '--------------------------------------------------------------------------
1400 GOTO 1580                                              ' Alloc=0:Decal=0
1420 '--------------------------------------------------------------------------
1430 '                                                        Alloc=1:Decal=0
1440 '                                                        Ajout fin index
1460 INDEX(LIGNE)=IX:CLE$(LIGNE)=NOM$
1470 GET #2,INDEX(LIGNE):LSET PT$(1)=MKI$(RANG):LSET SCLE$(1)=NOM$
1480 PUT #2,INDEX(LIGNE)
1490 GOSUB 890
1500 RETURN
1510 '--------------------------------------------------------------------------
1520 '                                                        Alloc=1:Decal=1
1530 FOR I=NCL-1 TO LIGNE+1 STEP-1:CLE$(I+1)=CLE$(I):INDEX(I+1)=INDEX(I):NEXT I
1540 INDEX(LIGNE+1)=IX
1550 GOTO 1580
1560 '-------------------------------------------------------------------------
1570 '                                                        Alloc=0:Decal=1
1580 GET #2,INDEX(LIGNE):T3$=PT$(NCL):T4$=SCLE$(NCL)
1590 FOR U=NCL-1 TO ISER STEP-1:LSET PT$(U+1)=PT$(U):LSET SCLE$(U+1)=SCLE$(U):NEXT
U
1600 LSET PT$(ISER)=MKI$(RANG):LSET SCLE$(ISER)=NOM$:PUT #2,INDEX(LIGNE):CLE$(LIGNE
)=SCLE$(1)
1620 '
1630 IF CVI(T3$)=0 THEN GOSUB 890:RETURN
1640 LIGNE=LIGNE+1:GET #2,INDEX(LIGNE)
1650 T1$=PT$(NCL):T2$=SCLE$(NCL)
1660 FOR U=NCL-1 TO 1 STEP-1:LSET PT$(U+1)=PT$(U):LSET SCLE$(U+1)=SCLE$(U):NEXT U
1670 LSET PT$(1)=T3$:LSET SCLE$(1)=T4$:PUT #2,INDEX(LIGNE):CLE$(LIGNE)=SCLE$(1)
1680 T3$=T1$:T4$=T2$:GOTO 1630
1690 '==========================================================================
1700  FOR I=1 TO NCL                                        ' LISTE TRIEE
1710    PRINT
1720    GET #2,INDEX(I)
1740    FOR J=1 TO NCL
1750       X=CVI(PT$(J)):IF X><0 THEN GET #1,X : PRINT NM$
1760     NEXT J
1780  NEXT I
1790 RETURN
1800 '===================================================== SUPPRESION D'UNE CLE
1820 PRINT:X$="":INPUT "CLE? ";X$:IF X$="" THEN RETURN
1830 NOM$=X$:GOSUB 960:ON R GOTO 1850,1840
1840 PRINT "N'EXISTE PAS ":PRINT:GOTO 1820
1850 PRINT NM$;:R$="":INPUT "ANNULE OK? (O) ";R$:IF R$<>"O" THEN GOTO 1820
1860 GOSUB 1880
1870 GOTO 1820
1880 '-------------
1890 FOR I=ISER TO NCL-1
1900   LSET SCLE$(I)=SCLE$(I+1):LSET PT$(I)=PT$(I+1)
1910 NEXT I
1920 LSET PT$(NCL)=MKI$(0)
1930 CLE$(LIGNE)=SCLE$(1)
1940 PUT #2,INDEX(LIGNE)
1950 IF ASC(SCLE$(1))=0 THEN FOR I=LIGNE TO NCL:CLE$(I)=CLE$(I+1):INDEX(I)=INDEX(I+
1):NEXT I
1960 GOSUB 890
1970 RETURN
```

```
1980 '=================================================================
1990 '                                   CREATION INDEX (SI DESTRUCTION)
2000 '    utilise 2 tables TC$() et INDEX()
2010 '
2020 KILL #2:OPEN "R",#2,"ITO"
2030 FOR I=1 TO NCL:CLE$(I)="":INDEX(I)=0:NEXT I
2040 GOSUB 800                 ' Appel initialisation index
2050 IND=0
2060 FOR S=1 TO LOF(1)
2070   GET #1,S:IF ASC(NM$)=0 THEN 2100
2080   PRINT NM$,S
2090   IND=IND+1:TC$(IND)=NM$:IX%(IND)=S
2100 NEXT S
2110 NB=IND:GOSUB 2290          ' Appel TRI
2120 '------------------- Sauvegarde index trie
2130 K=1
2140 FOR I=2 TO NCL
2150   GET #2,I
2160   FOR J=1 TO NCL
2170     LSET SCLE$(J)=TC$(K):LSET PT$(J)=MKI$(IX%(K))
2180     IF K=IND THEN PUT #2,I:CLE$(I-1)=SCLE$(1):INDEX(I-1)=I:GOTO 2250
2190     K=K+1
2200   NEXT J
2210   PUT #2,I
2220   CLE$(I-1)=SCLE$(1):INDEX(I-1)=I
2230 NEXT I
2240 '
2250 FOR I=1 TO NCL:LSET SCLE$(I)=CLE$(I):LSET PT$(I)=MKI$(INDEX(I)):NEXT I
2260 PUT #2,1
2270 RETURN
2280 '-----------------------------------------
2290 PRINT:PRINT "JE TRIE POUR VOUS NB=",NB:PRINT:ECART=NB
2300 '
2310 ECART=INT(ECART/2):IF ECART<1 THEN RETURN
2320 J=1:K=NB-ECART
2330 '
2340 I=J
2350 '
2360 L=I+ECART
2370 IF TC$(I)<TC$(L) GOTO 2410
2380 SWAP TC$(L),TC$(I):SWAP IX%(L),IX%(I):
2390 I=I-ECART:IF I<1 GOTO 2410 ELSE GOTO 2360
2400 '
2410 J=J+1:IF J>K GOTO 2310 ELSE GOTO 2340
```

ACCES INDEXE AVEC RECHERCHE DICHOTOMIQUE

Le temps de recherche séquentielle dans une table devient long dès que le nombre d'éléments est important (une seconde pour 200 éléments en Basic interprété).

Une façon de réduire le temps de recherche dans une table, pourvu qu'elle soit triée, consiste à procéder par dichotomie :

- Le principe de la recherche dichotomique consiste à comparer l'élément cherché à celui du milieu de la table. Selon qu'il est plus grand ou plus petit, on sélecte la moitié de la table où il se trouve
- En procédant de la même façon sur la moitié sélectée, on converge vite vers l'élément recherché

```
<---------- N nombres en ordre croissant ------->
!                                              !
3 6 7 .......     16 18 .....   40 50 60       70
!                  !                           !
INF               MIL                         SUP
```

```
--------------- Recherche dichotomique simple

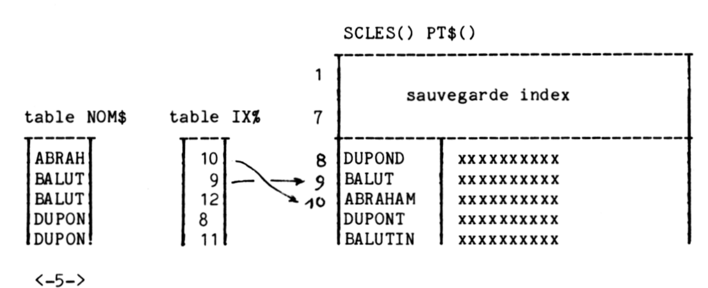
10 INPUT "Quel nombre cherchez vous? ";X
20 '
100 INF=1:SUP=N
105 '
110 IF INF>SUP THEN PRINT "Ce nombre n'existe pas":GOTO 10
115 MIL=INT(INF+SUP)/2)
130 IF X=A(MIL) THEN PRINT "Position de:";X;MIL:GOTO 10
140 IF X<A(MIL) THEN SUP=MIL-1:GOTO 110 ELSE INF=MIL+1:GOTO 110
```

Dès que la variable SUP devient inférieure à INF, nous en concluons que l'élément cherché n'existe pas.

Lorsque l'élément cherché n'existe pas, nous devons souvent l'insérer dans la table, et bien entendu, de façon à ce que la table soit toujours en ordre croissant. Au moment où nous détectons que l'élément cherché n'existe pas, la variable MIL indique soit la position d'insertion, soit cette position-1. Avant de quitter le sous-programme, il nous suffit, pour fournir la position d'insertion, de faire :

```
1080 IF INF>SUP THEN R=2:ISER=MIL:IF NOM$>CLE$(ISER) THEN ISER=ISER+1:RETURN
       ELSE RETURN
```

Autre difficulté : le même élément peut exister en plusieurs exemplaires. C'est par exemple le cas dans le programme étudié ou pour réduire la taille de la table des CLES, seules les premières lettres ont été stockées :



- Il convient de se positionner sur la première clé de la table CLE$() pour ensuite explorer la séquence des clés égales. Or la recherche dichotomique positionne sur une clé quelconque
- La façon la plus simple pour se positionner sur la première clé, consiste, au moment où on a retrouvé une clé, à comparer celle-ci à celle immédiatement inférieure. S'il y a égalité, nous considérons que la clé cherchée n'est pas trouvée et nous continuons la recherche.

```
1110 IF NOM$=CLE$(MIL) THEN
       IF CLE$(MIL)=CLE$(MIL-1) THEN SUP=MIL+1:GOTO SUITE ELSE Q=1:RETURN
```

Pour le programme INDD, la table d'index est sauvegardée dans les enregistrements de 1 à 7 du fichier principal. L'index doit être sauvegardé totalement à chaque ajout de clé puisque toute la table peut être modifiée par les décalages des clés. La régénération de l'index en cas d'incident sera faite ainsi :

1/ Constitution de CLE$() et INDEX() par lecture du fichier

2/ Tri de CLE$() et INDEX()

3/ Sauvegarde de ces tables

```
10 ' INDD 25.12.80
20 '
30 '       ACCES INDEXE AVEC RECHERCHE DICHOTOMIQUE
40 '       ========================================
50 '
70 '
80 '       NBC : Nombre de cles dans la table d'index
90 '
100 CLEAR(5000)
110 DEFINT A-Z                 ' NCL: Nombre de cles par enreg de l'index
120 LGCL=5:NCL=15                         ' Sur TRS80 NCL=(254/(5+2))
130 DIM SCLE$(NCL ),PT$(NCL)              ' SCLE$():PT$() Tables pour FIELD index
140 OPEN "R",1,"IND"
150 '
160 FIELD #1,10 AS NM$,8 AS PREN$,20 AS TEL$
170 FOR I=1 TO NCL:FIELD #1,(LGCL+2)*(I-1) AS D$,(LGCL) AS SCLE$(I),2 AS PT$(I):NEX
T I  ' Pour FIELD index
180 '
190 DIM CLE$(500),INDEX(500)
200 '
210 GET #1,1:IF ASC(SCLE$(1))=0 THEN GOSUB 870
220 GOSUB 530                              ' Appel lecture index
230 PRINT TAB(20) "Modes:":PRINT
240 PRINT TAB(30) "C : Creation"
250 PRINT
260 PRINT:INPUT "      Mode? (C,RI,FIN) ";M$
270 IF M$="C" THEN GOSUB  320
280 IF M$="L" THEN GOSUB 1170
290 IF M$="FIN" THEN CLOSE #1:STOP
300 GOTO 260
310 '======================================================================
320 '                                                      AJOUT D'UN NOM
330 PRINT:INPUT "Nom ? ";NOM$
340 GOSUB 750:ON R  GOTO 320,350          ' Appel recherche cle
350 RANG=LOF(1):GET #1,RANG               ' LOF(1)+1 sur TRS80
360 LSET NM$=NOM$
370 INPUT "Prenom?    ";X$:LSET PREN$=X$
380 INPUT "Telephone? ";X$:LSET TEL$=X$
390 PRINT :PRINT TAB(30):R$="":INPUT "OK? ";R$:IF R$><"O" GOTO 320
400 PUT #1,RANG
410 GOSUB 440
420 RETURN
430 '----------------------------------------------------------------------
440 IF NBC=0  OR ISER>NBC THEN GOTO 480         ' INSERTION D'UNE  CLE
450 '
460 FOR K=NBC TO ISER STEP-1:CLE$(K+1)=CLE$(K):INDEX(K+1)=INDEX(K):NEXT K
470 '
480 CLE$(ISER)=NOM$:INDEX(ISER)=RANG:NBC=NBC+1
490 GOSUB 920                                   ' Appel sauvegarde index
500 RETURN
510 '======================================================================
520 '                    LECTURE DE L'INDEX DISQUE DANS CLE$() et INDEX()
530 NBC=0                    ' NBC:Nombre de cles
540 FOR I=1 TO 7
550     GET #1,I
560     FOR J=1 TO NCL
570        IF ASC(SCLE$(J))=0 GOTO 600
580        NBC=NBC+1:CLE$(NBC)=SCLE$(J):INDEX(NBC)=CVI(PT$(J))
590     NEXT J
600 NEXT I
610 RETURN
620 '
```

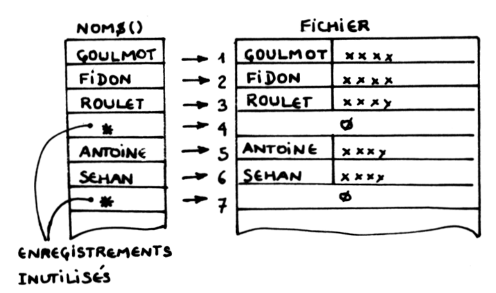
```
690 '============================================================================
700 '                                              RECHERCHE D'UNE CLE
710 '
720 '          Retour:   R=1 : la cle existe / R=2 : la cle n'existe pas
730 '
740 '
750 L=LEN(NOM$)                    ' L'operateur n'entre que les premieres lettres
760 '
770 GOSUB 1050                     ' Appel recherche dicho
780 ON R GOTO 810,790
790 RETURN                         ' Le NOM n'est pas trouve
800 '
810 GET #1,INDEX(ISER)                   ' Une cle a ete trouvee dans CLE$()
820 IF NOM$=LEFT$(NM$,L) THEN RANG=INDEX(I):R=1:RETURN
830 IF NOM$<NM$ OR INDEX(ISER)=0 THEN R=2:RETURN
840 ISER=ISER+1:GOTO 810
850 '-----------------------------------------------------------------------
860 '
870 PRINT :PRINT "INITIALISATION INDEX":PRINT
880 FOR I=1 TO 7:GET #1,I:PUT #1,I:NEXT I
890 RETURN
900 '-----------------------------------------------------------------------
910 '                          SAUVEGARDE de CLE$() et INDEX() dans enreg 1 a 7
920 W=1
930  FOR I=1 TO 7
940    GET #1,I
950     FOR J=1 TO NCL
960        IF CLE$(W)="" THEN PUT #1,I:RETURN       ' Derniere cle?
970        LSET SCLE$(J)=CLE$(W):LSET PT$(J)=MKI$(INDEX(W)):W=W+1
980     NEXT J
990 PUT #1,I
1000 NEXT I
1010 '-----------------------------------------------------------------------
1020 '                                                  RECHERCHE DICHOTOMIQUE
1030 ' INF:borne inferieure/ SUP:superieure/ MI:milieu /ISER:position d'insertion
1040 '             Retour: R=1 : cle trouvee  / R=2 : cle non trouvee
1050 INF=1:SUP=NBC
1060 IF SUP<1 THEN R=2:ISER=1:RETURN
1070 '
1080 IF INF>SUP THEN R=2:ISER=MI:IF NOM$>CLE$(ISER) THEN ISER=ISER+1:RETURN ELSE RE
TURN
1090 '
1100  MIL=INT((INF+SUP)/2)
1110 IF NOM$=LEFT$(CLE$(MIL),L)  OR CLE$(MIL)=LEFT$(NOM$,LGCL) THEN
        IF (NOM$=LEFT$(CLE$(MIL-1),L) OR CLE$(MIL-1)=LEFT$(NOM$,LGCL)) THEN
          SUP=MIL-1:GOTO 1080 ELSE R=1:ISER=MIL:RETURN
1120 '
1130 IF NOM$<CLE$(MIL) THEN SUP=MIL-1:GOTO 1080 ELSE INF=MIL+1:GOTO 1080
1140 '
1150 '-------------------------------------------------------------------
1160 '                                               LISTE TRIEE(par l'index)
1170 PRINT
1180 FOR I=1 TO NBC
1190   GET #1,INDEX(I)
1200   PRINT NM$,PREN$,TEL$
1210 NEXT I
1220 RETURN
```

```
10 ' dicho
20 '                RECHERCHE DICHOTOMIQUE ET INSERTION
30 '
40 DIM A(100)                   ' 100 elements maxi pour table A()
50 NBE=0                        ' Nb d'elements de A()
60 '
70 INPUT "Quel element cherchez vous? ";X
80 GOSUB 260:ON R GOTO 100,130
90 '
100 PRINT "Element place en:";MIL:GOTO 70
110 '
120 '------ Insertion
130 PRINT "Insertion en:";ISERT
140 IF ISERT>NBE THEN 190
150 FOR I=NBE TO ISERT STEP-1       ' Decalages
160   A(I+1)=A(I)
170 NEXT I
180 '
190 A(ISERT)=X                      ' Insertion
200 NBE=NBE+1
210 PRINT:FOR I=1 TO NBE:PRINT I,A(I):NEXT I
220 GOTO 70
230 '------------------------------------ Recherche dichotomique
240 '     Retour: R=1  Element trouve    R=2 Element non trouve
250 '             ISERT: Position d'insertion
260 INF=1:SUP=NBE
270 IF NBE<1 THEN ISERT=1:R=2:RETURN                  ' Y'a rien dans la table
280 '
290 IF INF>SUP THEN R=2:ISERT=MIL:IF X>A(ISERT) THEN ISERT=ISERT+1:RETURN ELSE RETU
RN
300 MIL=INT((INF+SUP)/2)
310 IF X=A(MIL) THEN R=1:RETURN                      ' Element trouve
320 IF X>A(MIL) THEN INF=MIL+1:GOTO 290 ELSE SUP=MIL-1:GOTO 290
330 '
340 ' RUN
350 ' Quel element cherchez vous? 6
360 ' Insertion en: 1
370 '
380 ' Quel Element cherchez vous? 10
390 ' Insertion en: 2
400 '
410 ' Quel element cherchez vous? 7
420 ' Insertion en: 2
```

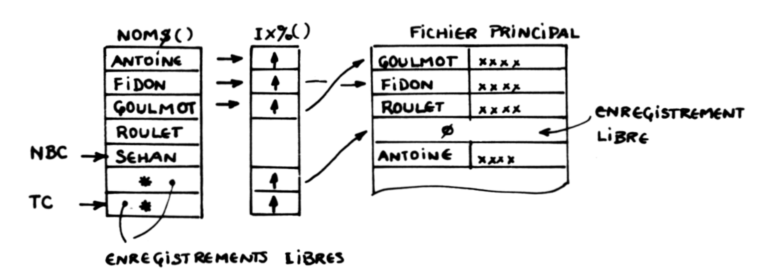
ACCES INDEXE ET ALLOCATION DYNAMIQUE

Nous connaissons le problème de la récupération des enregistrements supprimés dans un fichier. Une réorganisation de celui-ci la permet, mais il est plus élégant de les récupérer dynamiquement grâce à des bits-map ou des chaînages.

Dans le cas d'un accès indexé, il suffit de repérer les enregistrements inutilisés par un caractère quelconque ('*' par exemple) dans la table d'index. C'est la méthode que nous avons utilisée dans le programme EDIR.



Si la table a été rangée en ordre croissant, dans le but d'y effectuer une recherche dichotomique, les pointeurs vers les enregistrements libres seront placés en fin de table afin de faciliter la recherche dichotomique.

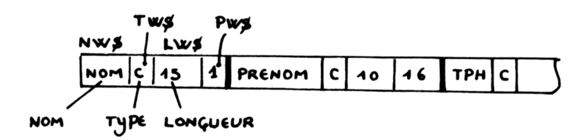


Les pointeurs NBC et TC n'ont pas à être sauvegardés sur disque. Leurs valeurs seront calculées au moment où la table NOM$() (sauvegardée sur disque) est transférée en mémoire centrale en début de session.

EDITEUR DE FICHIER RANDOM

Ce programme permet de gérer des fichiers Random sans qu'il soit nécessaire d'écrire des programmes. L'opérateur décrit les différentes zones de chaque enregistrement (*NOM, LONGUEUR, TYPE)* Ces caractéristiques sont stockées dans l'enregistrement numéro 1 du fichier.

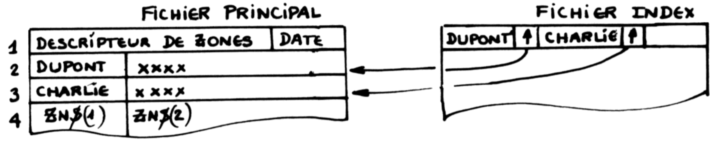
Une saisie d'écran caractère par caractère est générée en fonction de ces zones.



* DESCRIPTEUR DE ZONES (ENREGISTREMENT N°1)

D'autres fonctions sont également assurées :

- tri selon une zone quelconque
- tri multicritères
- édition automatique
- accès indexé sur la première zone du fichier





Exemple : Création d'un fichier avec trois zones (*NOM, PRENOM, TELEPHONE*)

```
Nom de zone?   NOM
Type(C,S,N)?   C
Longueur   ?   15

Nom de zone?   PRENOM
Type(C,S,N)?   c
Longueur   ?   12

Nom de zone?   TPH
Type       ?   C
Longueur   ?   22

Nom de zone?   <ENTER>
```

```
10 CLS:PRINT "EDIR 9.1.81   EDITEUR DE FICHIER RANDOM "
20 CLEAR(4000):DEFINT I-S
30 '
40 '  PERMET DE GERER UN FICHIER 'RANDOM' SANS ECRIRE DE PROGRAMME
50 '
60 '       .GENERATION D'UNE SAISIE D'ECRAN
70 '       .EDITIONS
80 '       .TRI MULTICRITERES
90 '       .ACCES INDEXE
100 '
110 DIM NW$(15),PW$(15),LW$(15),TW$(15),LZ(15),TZ(15)
120 DIM PZ(15),TZ$(15),LS(15),NZ$(15),Z$(15),ZN$(15),ID(15),Y(15),X(15)
130 DIM R$(15),CLE$(300),IX%(300)
150 '  NZ : Nombre de zones par enregistrement
160 '  NW$() : NZ$() : Noms des zones symboliques
170 '  TW$() : TZ () : Types des zones  (C,I,S,N)
180 '  LW$() : LZ () : Longueur des zones fichier
200 '  PW$() : PZ () : Position des zones dans l'enregistrement
210 '
220 PRINT:PRINT:PRINT "   Nom du fichier (XX:0) ";:LINE INPUT NF$
230 IF NF$><"" THEN OPEN "R",1,NF$ ELSE GOTO 220
240 OPEN "R",#2,"I"+NF$                 ' Ouverture fichier index
250 LCL=8:NCL=INT(254/(LCL+2))          ' LCL:Longueur cles
255 DIM CTERE$(NCL),PT$(NCL)
260 FOR I=1 TO NCL:
    FIELD #2,(LCL+2)*(I-1) AS D$,(LCL) AS CTERE$(I),2 AS PT$(I):NEXT I
270 GOSUB 2230                          ' Appel lecture fichier index
280 FIELD #1,195 AS D$,1 AS J$,1 AS M$,1 AS A$:GET #1,1  ' Field date
290 DT$=STR$(ASC(J$))+STR$(ASC(M$))+STR$(ASC(A$))
300 PRINT :PRINT "DATE: ";DT$
310 X=0:INPUT "JOUR  ";X:IF X<32 AND X>0 THEN LSET J$=CHR$(X)
    ELSE GOTO 380
320 X=0:INPUT "MOIS  ";X:IF X><0 THEN LSET M$=CHR$(X) ELSE GOTO 340
330 X=0:INPUT "AN    ";X:IF X<90 AND X>0 THEN LSET A$=CHR$(X)
340 PUT #1,1
350 DT$=STR$(ASC(J$))+STR$(ASC(M$))+STR$(ASC(A$))   ' Date fichier
380 GOSUB 690:ON Q GOTO 400,390 ' Lecture descripteur de zones
390 GOSUB 890:GOTO 380          ' Introduction descripteur de zones
400 IF NZ<1 THEN PRINT :PRINT "Declarez 2 zones au moins":PRINT:GOTO 390
410 FOR I=1 TO NZ               ' Field pour fichier principal
420   FIELD #1,PZ(I)-1 AS D$,LZ(I) AS ZN$(I):NEXT I
435 PRINT:INPUT "APPUYER SUR <ENTER> ";R$
440 CLS:PRINT "GESTION DU FICHIER: ";NF$:PRINT "FONCTIONS POSSIBLES.."
455 '-------------------------------- MENU
460 PRINT TAB(20) "ME  : AJOUT ET MODIF PAR NO ENREGISTREMENT"
470 PRINT TAB(20) "LF  : LISTE DU FICHIER NON TRIEE"
480 PRINT TAB(20) "LFT : LISTE DU FICHIER TRIE"
490 PRINT TAB(20) "MC  : AJOUT ET MODIF PAR CLE"
500 PRINT TAB(20) "SU  : SUPPRESSION PAR CLE"
510 PRINT TAB(20) "CRI : CREATION INDEX (SI INCIDENT)"
520 PRINT TAB(20) "FIN : FIN DE SESSION (OBLIGATOIRE)"
530 PRINT:INPUT "MODE CHOISI (ME,MC,SU,LF,LFT,DZ,FIN,..)";M$
540 IF M$="ME" THEN GOSUB 2170
550 IF M$="LF" THEN GOSUB 1340
560 IF M$="LFT" THEN GOSUB 2630
570 IF M$="FIN" THEN GOSUB 4060:CLOSE #1,#2:CMD "S"SYSTEM
580 IF M$="CRI" THEN GOSUB 3890
590 IF M$="MC" THEN GOSUB 4180
600 IF M$="SU" THEN GOSUB 2460
610 IF M$="DZ" THEN GOSUB 890
620 GOTO 440
630 '==========================================================
```

```
670 '                          AFFICHAGE DU DESCRIPTEUR DE ZONES
680 '
690 GOSUB 860:GET #1,1:IF ASC(NW$(1))=0 THEN Q=2:RETURN
700 Z=0:PRINT
710 CLS:PRINT "RECAPITULATIF DE LA DESCRIPTION DES ZONES ":PRINT
720 PRINT "NOM              POSITION      LONGUEUR      TYPE":PRINT
730 FOR I=1 TO 15
740   IF ASC(NW$(I))=0 THEN NZ=I-1:GOTO 830
750   NZ$(I)=NW$(I):LZ(I)=ASC(LW$(I)):LS(I)=LZ(I):PZ(I)=ASC(PW$(I)):
      TZ$(I)=TW$(I)
760  TZ(I)=1
770  IF TW$(I)="S" THEN TZ(I)=2:LS(I)=7 ELSE TZ(I)=1    ' LS() LG SAISIE
780  IF TW$(I)="N" THEN TZ(I)=4:LS(I)=3
790  IF TW$(I)="E" THEN TZ(I)=3:LS(I)=5
800 PRINT NZ$(I),PZ(I),ASC(LW$(I)),TZ$(I)
810 NEXT I
820 NZ=I-1
830 Q=1:RETURN
840 '----------------------------------------------------------------
850 '                                  FIELD DU DESCRIPTEUR DE ZONE
860 FOR Q=1 TO 15:FIELD #1,(Q-1)*10 AS D$,7 AS NW$(Q),1 AS PW$(Q),
1 AS LW$(Q),1 AS TW$(Q):NEXT Q:RETURN
870 '----------------------------------------------------------------
880 '                    INTRODUCTION DE LA    DEFINITION DES ZONES
890 GET #1,LOF(1)+1:PRINT:PRINT "DEFINISSEZ LES ZONES ! "
900 PRINT:PRINT "NOM DE ZONE : 7 CARACTERES MAXI([ POUR ANNULER)":PRINT
910 PRINT "TYPE DE LA ZONE: ENTRER       C        POUR CHAINES DE CARACT"
920 PRINT"                  ENTRER       N        POUR NUMERIQUE <256"
930 PRINT"                  ENTRER       E        POUR ENTIERS (<32000)
940 PRINT"                  ENTRER       S        POUR SIMPLE PRECISION"
950 PRINT :PZ=1
960 FOR I=1 TO 15
970   PRINT:PRINT "NOM DE ZONE  ";I;
980   X$="":INPUT X$:IF X$="" GOTO 1090
990   IF X$="[" AND I<>1 THEN I=I-1:X$="":GOTO 970
1000   LSET NW$(I)=X$
1010   LSET PW$(I)=CHR$(PZ)
1020   X$="":INPUT"TYPE (C,N,E,S)  ";X$:LSET TW$(I)=X$
1030   IF X$="S" THEN X$="4":GOTO 1070
1040   IF X$="N" THEN X$="1":GOTO 1070
1050   IF X$="E" THEN X$="2":GOTO 1070
1060   X$="":INPUT "LONGUEUR ZONE   ";X$:IF VAL(X$)=0 GOTO 1060
1070   LSET LW$(I)=CHR$(VAL(X$)):PZ=PZ+VAL(X$)
1080 NEXT I
1090 PUT #1,1
1100 CLOSE #2:KILL "I"+NF$:OPEN "R",#2,"I"+NF$
1110 NBC=0:FOR I=1 TO 300:CLE$(I)="":IX%(I)=0:NEXT I
1120 RETURN
1130 '
1140 '===============================================================
1150 '                     SAUVEGARDE DE LA PARTIE D'INDEX MODIFIEE
1160 DB=INT((NI-1)/NCL)      ' NI:no element de l'index modifie
1170 K=DB*NCL+1
1180 GET #2,DB+1
1190 FOR J=1 TO NCL
1200   IF CLE$(K)="" THEN PUT #2,DB+1:RETURN
1210   LSET CTERE$(J)=CLE$(K):LSET PT$(J)=MKI$(IX%(K)):K=K+1
1220 NEXT J
1230 PUT #2,DB+1
1240 RETURN
1250 '
```

```
1320 '=============================================================
1330 '                                        EDITIONS FICHIER
1340 PRINT:PRINT:GOSUB 1990:PRINT          ' Appel quelles zones?
1350 INPUT "COMMENTAIRE ";XX$
1360 E=1:PRINT:R$="":INPUT "EDITION IMPRIMANTE (O/N) ";R$:
     IF R$="O" THEN E=2
1370 GOSUB 2530                            ' Appel quelles zones?
1380 ON E GOSUB 1390,1500:GOTO 1610
1385 '------------------------- Edition titre ecran
1390 PRINT:PRINT "LISTE DU FICHIER  ";VF$;"     ";NF$:PRINT:PRINT
1410 PRINT "     ";
1420 FOR W=1 TO 15
1430   IF Z$(W)="" GOTO 1470
1440   PRINT NZ$(ID(W));                    ' ID(): No de zone
1450   FOR K=7 TO LZ(ID(W)):PRINT " ";:NEXT K
1460 NEXT W
1470 PRINT:PRINT
1480 RETURN
1490 '---------------- Edition titre imprimante
1500 LPRINT:LPRINT "      LISTE DU FICHIER:  ";"'";NF$;"'";
     "    ";DT$;"      ";XX$
1510 LPRINT:LPRINT
1520 LPRINT "    ";
1530 FOR W=1 TO 15                        ' Impression des noms de zones
1540    IF Z$(W)="" GOTO 1580
1550    LPRINT NZ$(ID(W));
1560    FOR K=7 TO LZ(ID(W)):LPRINT " ";:NEXT K
1570 NEXT W
1580 LPRINT:LPRINT
1590 RETURN
1600 '---------------- Lecture du fichier
1610 FOR C=L1 TO L2
1630   GET #1,C:IF ASC(Z$(1))=0 GOTO 1650
1640   ON E GOSUB 1840,1710
1650 NEXT C
1660 IF E=2 THEN LPRINT CHR$(12)
1670 PRINT:INPUT"FAIRE <ENTER> POUR CONTINUER";KK$
1680 RETURN
1690 '------------------------ Edition imprimante
1710 LPRINT USING "### ";C;
1720 FOR W=1 TO 15
1730    IF Z$(W)="" GOTO 1800
1740    ON TZ(ID(W)) GOTO 1780,1760,1750,1770
1750    LPRINT USING " ##### ";CVI(Z$(W));:LPRINT " ";:GOTO 1790
1760    LPRINT USING "#######";CVS(Z$(W));:LPRINT " ";:GOTO 1790
1770    LPRINT USING " ####  ";ASC(Z$(W));:LPRINT " ";:GOTO 1790
1780    IF LEN(Z$(W))>=7 THEN LPRINT Z$(W);" ";
         ELSE LPRINT USING "%      %";Z$(W);
1790 NEXT W
1800 LPRINT
1810 RETURN
1820 '------------------------  Edition ecran
1840 PRINT USING "#### ";C;
1850 FOR W=1 TO 15
1860   IF Z$(W)="" THEN GOTO 1930
1870   ON TZ(ID(W)) GOTO 1910,1890,1880,1900
1880   PRINT USING " ##### ";CVI(Z$(W));:PRINT " ";:GOTO 1920
1890   PRINT USING "#######";CVS(Z$(W));:PRINT " ";:GOTO 1920
1900   PRINT USING " ###   ";ASC(Z$(W));:PRINT " ";:GOTO 1920
1910   IF LEN(Z$(W))>=7 THEN PRINT Z$(W);" ";
        ELSE PRINT USING "%      %";Z$(W);
1920 NEXT W
1930 PRINT:RETURN
```

```
1970 '-----------------------------------------------------------
1980 '                              QUELLES ZONES A EDITER?
1990 GOSUB 690:PRINT
2000 Z=0                                  ' ID():No de zones a editer
2010 FOR I=1 TO 15:Z$(I)="":NEXT I
2040 PRINT:PRINT "ZONE: ";
2050 PRINT TAB(12) :FOR I=1 TO 6 :PRINT NZ$(I);" ";:NEXT I:PRINT
2070 PRINT TAB(12):FOR I=7 TO 15:PRINT NZ$(I);" ";:NEXT I:PRINT:PRINT
2090 X$="":INPUT "ZONE A IMPRIMER: ";X$:IF X$="" THEN RETURN
2100 FOR I=1 TO 15
2110   IF LEFT$(NZ$(I),LEN(X$))=X$ THEN Z=Z+1:ID(Z)=I:GOTO 2140
2120 NEXT I
2130 GOTO 2090
2135 '
2140 FIELD #1,PZ(I)-1 AS D$,LZ(I) AS Z$(Z):GOTO 2090
2160 '============================================================
2170 PRINT:NO=0:PRINT :PRINT "FIN FICHIER=";LOF(1);:
     INPUT "  NO ENREGISTREMENT ";NO:
2180 IF NO<2 THEN RETURN
2190 GET #1,NO:PRINT
2200 GOSUB 3250:PUT #1,NO:GOTO 2170
2210 '============================================================
2220 '                                     LECTURE DE L'INDEX
2230 NBC=0
2240 FOR I=1 TO 15
2250   GET #2,I
2260   FOR J=1 TO NCL
2270     IF ASC(CTERE$(J))=0 GOTO 2300
2280     NBC=NBC+1:CLE$(NBC)=CTERE$(J):IX%(NBC)=CVI(PT$(J))
2290   NEXT J
2300 NEXT I
2310 RETURN
2320 '============================================================
2330 ECART=NB:PRINT:PRINT "JE TRIE POUR VOUS":PRINT      ' TRI
2340 '
2350 ECART=INT(ECART/2):IF ECART<1 THEN RETURN
2360 J=1:K=NB-ECART
2370 I=J
2380 L=I+ECART
2390 IF CLE$(I)<CLE$(L) GOTO 2430
2400 X$=CLE$(L):CLE$(L)=CLE$(I):CLE$(I)=X$:
           X=IX%(L):IX%(L)=IX%(I):IX%(I)=X
2410 I=I-ECART:IF I<1 GOTO 2430 ELSE GOTO 2380
2420 '
2430 J=J+1:IF J>K GOTO 2350 ELSE GOTO 2370
2440 '============================================================
2450 '                          SUPPRESSION(* POUR ENREG SUPPRIME)
2460 PRINT:X$="":INPUT "CLE ";X$:IF X$="" THEN RETURN
2470 GOSUB 4340:ON Q GOTO 2490,2480      ' Appel recherche cle
2480 PRINT :PRINT "N'existe pas ":PRINT:GOTO 2460
2490 PRINT:PRINT ZN$(1);:R$="":INPUT "Annule OK (O) ";R$:
     IF R$><"O" GOTO 2460
2500 CLE$(PS)="*":GET #1,LOF(1)+1:LSET ZN$(1)="*":
     PUT #1,IX%(PS):NI=PS:GOSUB 1160    ' Appel sauvegarde cle$()
2510 GOTO 2460
2520 '============================================================
2530 INPUT"NO D'ENREGISTREMENT DE DEBUT ";L1$  ' Quelles bornes?
2540 IF L1$="" THEN L1=2 ELSE L1=VAL(L1$)+1
2550 INPUT"NO D'ENREGISTREMENT DE FIN   ";L2$
2560 IF L2$="" THEN L2=LOF(1) ELSE L2=VAL(L2$)+1
2570 RETURN
```

```
2580 '==============================================================
2600 '
2610 '
2620 '                                        TRI MULTICRITERES
2630 NB=0:GOSUB 690
2640 GOSUB 4060                      ' Sauvegarde de la table d'index
2650 FOR I=1 TO 4:T(I)=0:NEXT I:PRINT
2660 X$="":INPUT "PREMIER CRITERE DE TRI :";X$:IF X$="" THEN RETURN
2670 FOR I=1 TO 15
2680   IF LEFT$(NZ$(I),LEN(X$))=X$ THEN T(1)=TZ(I):GOTO 2710
2690 NEXT I
2700 GOTO 2660
2710 FIELD #1,PZ(I)-1 AS D$,LZ(I) AS CRIT$(1)
2720 FOR K=2 TO 3
2730    PRINT:X$="":PRINT K;"CRITERE DE TRI : ";:INPUT X$:
        IF X$="" GOTO 2800
2740    FOR L=1 TO 15
2750      IF LEFT$(NZ$(L),LEN(X$))=X$ THEN T(K)=TZ(L):GOTO 2780
2760    NEXT L
2770    GOTO 2730
2780 FIELD #1,PZ(L)-1 AS D$,LZ(L) AS CRIT$(K)
2790 NEXT K
2800 GOSUB 2000                        ' Appel quelles zones?
2810 E=1:PRINT:R$="":INPUT "EDITION IMPRIMANTE (O/N) ";R$:
IF R$="O" THEN E=2:INPUT"COMMENTAIRE ";XX$
2820 GOSUB 2530:PRINT                   ' Appel calcul bornes L1 ET L2
2830 '------------ Lecture du fichier
2840 FOR I=L1 TO L2
2850   GET #1,I
2860   ON T(1) GOTO 2870,2880,2890,2900
2870   IF ASC(CRIT$(1))=0 GOTO 3000
        ELSE : NB=NB+1:CLE$(NB)=CRIT$(1):GOTO 2910
2880   IF CVS(CRIT$(1))=0 GOTO 3000 ELSENB=NB+1:
       CLE$(NB)=RIGHT$("        "+STR$(CVS(CRIT$(1))),7):GOTO 2910
2890   IF CVI(CRIT$(1))=0 GOTO 3000 ELSENB=NB+1:
       CLE$(NB)=RIGHT$("        "+STR$(CVI(CRIT$(1))),7):GOTO 2910
2900   IF ASC(CRIT$(1))=0 GOTO 3000 ELSE
       NB=NB+1:CLE$(NB)=RIGHT$("    "+STR$(ASC(CRIT$(1))),5):GOTO 2910
2910   FOR K=2 TO 3
2920     IF T(K)><0 THEN ON T(K) GOTO 2930,2940,2950,2960 ELSE GOTO 2980
2930     IF ASC(CRIT$(K))=0 THEN GOTO 2980
          ELSE CLE$(NB)=CLE$(NB)+CRIT$(K):GOTO 2970
2940     IF CVS(CRIT$(K))=0 GOTO 2980
          ELSE CLE$(NB)=CLE$(NB)+RIGHT$(STR$(CVS(CRIT$(K))),7):GOTO 2970
2950     IF CVI(CRIT$(K))=0 GOTO 2980 ELSE CLE$(NB)=
         CLE$(NB)+RIGHT$("        "+STR$(CVI(CRIT$(K))),7):GOTO 2970
2960     IF ASC(CRIT$(K))=0 GOTO 2980 ELSE
         CLE$(NB)=CLE$(NB)+RIGHT$("    "+STR$(ASC(CRIT$(K))),5):GOTO 2970
2970   NEXT K
2980   IX%(NB)=I
2990   PRINT CLE$(NB)
3000 NEXT I
3010 GOSUB 2330                        ' Appel TRI
3020 '------------------------ Editions
3040 ON E GOSUB 1390,1500
3050 FOR I=1 TO NB
3060   GET #1,IX%(I)
3070   C=IX%(I):ON E GOSUB 1840,1710
3080 NEXT I
3090 IF E=2 THEN LPRINT CHR$(12)
3100 PRINT :INPUT "FAIRE <ENTER> POUR CONTINUER";KK$
3110 GOSUB 2230                          ' Lecture DE l'index
3120 RETURN
```

```
3190 '   TABLES UTILISEES : NZ$() : NOMS DES ZONES
3200 '                      LS () : LONGUEURS DES ZONES
3220 '
3230 '   SAISIE ECRAN      / Les coordonnees d'affichage sont calculees
3240 '
3250 FOR I=1 TO NZ:TR$(I)="":NEXT I
3260 CLS:GOSUB 3750
3270 PRINT@900,"ENREG:";NO:PRINT @924," [ POUR ZONES ARRIERES"
3290 P=1
3300 '------------- Saisie de N zones
3310 SET(35*2,P*3+1):GOSUB 3460              ' Appel saisie ligne
3320 RESET(35*2,P*3+1)                       ' Extinction curseur ligne
3330 ON R GOTO 3370,3400,3340    ' R=1:OK /R=2:RC /R=3:on remonte
3340 '
3350 IF P>1 THEN ON TZ(P) GOSUB 3600,3610,3610,3610:P=P-1:GOTO 3310
     ELSE 3310
3360 '
3370 ON TZ(P) GOSUB 3650,3670,3660,3680     ' Rangement de LIGNE$
3380 TR$(P)=LIGNE$
3390 '
3400 IF ASC(ZN$(P))=0 THEN IF TZ(P)=1 THEN LSET ZN$(P)="."
3410 PRINT @X,"";
3420 ON TZ(P) GOSUB 3600,3610,3610,3610     ' Reaffichage zone
3430 IF P=>NZ THEN :RETURN
3440 P=P+1:GOTO 3310
3450 '----------------- Saisie d'une ligne
3460 LIGNE$=""
3470 X=P*64+0+36:IF TZNE(P)=1 THEN AC$="." ELSE AC$="-"
3480 PRINT @X,"";
3490 '
3500 C$=INKEY$:IF C$="" THEN 3500           ' Attente d'un caractere
3510 C=ASC(C$):L=LEN(LIGNE$)
3520 IF C=13 THEN IF LIGNE$<>"" THEN R=1:RETURN ELSE R=2:RETURN
3530 IF C=91 THEN R=3:RETURN
3540 IF C=8 THEN IF L>0 THEN   PRINT CHR$(8);AC$;:
PRINT @X+L-1,"";:LIGNE$=LEFT$(LIGNE$,L-1):GOTO 3500
3550 ON TZNE(P) GOSUB 3620,3630,3630,3630:ON R GOTO 3560,3500
3560 PRINT C$;
3570 LIGNE$=LIGNE$+C$:IF L+1=>LS(P) THEN R=1:RETURN
3580 GOTO 3490
3590 '----------
3600 PRINT TR$(P);STRING$(LS(P)-LEN(TR$(P)),".");:RETURN
3610 PRINT TR$(P);STRING$(LS(P)-LEN(TR$(P)),"-");:RETURN
3620 IF C<32 THEN R=2:RETURN ELSE R=1:RETURN
3630 IF C >47 AND C<58 OR  C=46 THEN R=1:RETURN ELSE R=2:RETURN
3640 '
3650 LSET ZN$(P)=LIGNE$:RETURN
3660 LSET ZN$(P)=MKI$(VAL(LIGNE$)):RETURN
3670 LSET ZN$(P)=MKS$(VAL(LIGNE$)):RETURN
3680 LSET ZN$(P)=CHR$(VAL(LIGNE$)):RETURN
3690 '
3700 PRINT ZN$(P);:RETURN              ' Affichage ancienne valeur
3710 PRINT CVS(ZN$(P));:RETURN
3720 PRINT CVI(ZN$(P));:RETURN
3730 PRINT ASC(ZN$(P));:RETURN
3740 '----------------------- Affichage grille de saisie
3750 FOR P=1 TO NZ
3760   X=P*64:PRINT @X,NZ$(P);TAB(7) ":";
3770   PRINT @X+10,"";:ON TZ(P) GOSUB 3700,3710,3720,3730
3780   PRINT @X+36,"";
3790   ON TZ(P) GOSUB 3600,3610,3610,3610
3800 NEXT P
3810 RETURN
```

```
3870 '==========================================================
3880 '                              CREATION INDEX ZONE 1 (si incident)
3890 CLOSE #2:KILL "I"+NF$:OPEN "R",2,"I"+NF$
3900 FOR I=1 TO 300:CLE$(I)="":NEXT I:PRINT:PRINT
3910 '
3920 '
3930 '
3940 NBC=0                              ' NBC:nb de cles
3950 '
3960 FOR I=2 TO LOF(1)
3970   GET #1,I
3980   IF ASC(ZN$(1))=0 THEN LSET ZN$(1)="*"    ' Enreg vide?
3990   PRINT I,ZN$(1)
4000   NBC=NBC+1:CLE$(NBC)=ZN$(1):IX%(NBC)=I
4010 NEXT I
4020 GOSUB 4060                         ' Appel sauvegarde de cle$()
4030 RETURN
4040 '------------------- Sauvegarde de cle$() et ix%()
4060 W=1
4070 FOR I=1 TO 15
4080   GET #2,I
4090   FOR J=1 TO NCL       ' NCL=INT(254/(8+2))
4100     IF W>NBC  THEN PUT #2,I:RETURN
4110     LSET CTERE$(J)=CLE$(W):LSET PT$(J)=MKI$(IX%(W)):W=W+1
4120   NEXT J
4130 PUT #2,I
4140 NEXT I
4150 RETURN
4160 '==========================================================
4170 '                                          ACCES PAR CLE
4180 PRINT:PRINT:X$="":INPUT "CLE ";X$:IF X$="" THEN RETURN
4190 GOSUB 4340:ON Q GOTO 4200,4230
4200 PRINT:NO=RANG:GOSUB 3250:PUT #1,RANG    ' Q=1:Cle trouvee
4210 GOTO 4180
4220 '------------ Nouvelle cle
4230 PRINT:R$="":INPUT "NOUVELLE CLE (O) ";R$:IF R$><"O" THEN 4180
4240 FOR L=1 TO 300
4250   IF CLE$(L)><"" THEN IF ASC(CLE$(L))=42 THEN PS=L:
        RANG=IX%(L):GET #1,RANG:NO=RANG:
       LSET ZN$(1)=X$:GOSUB 3250:PUT #1,RANG:
       CLE$(PS)=ZN$(1):NI=PS:GOSUB 1160:GOTO 4180
4260   IF CLE$(L)="" GOTO 4280
4270 NEXT L
4280 RANG=LOF(1)+1:PRINT:GET #1,RANG:NO=RANG:LSET ZN$(1)=X$:GOSUB 3250
4290 PUT #1,RANG
4300 NBC=NBC+1:CLE$(NBC)=ZN$(1):IX%(NBC)=RANG:NI=NBC:GOSUB 1160
4310 GOTO 4180
4320 '------------------------------------ Recherche cle
4330 '
4340 W=1:L=LEN(X$)              ' Q=1:LA CLE EXISTE /Q=2: N'EXISTE PAS
4350 FOR I=W TO 300
4360   IF CLE$(I)="" THEN Q=2:RETURN
4370   IF X$=LEFT$(CLE$(I),L) OR CLE$(I)=LEFT$(X$,LCL) GOTO 4410
4380 NEXT I
4390 PRINT "INDEX PLEIN":STOP
4400 '
4410 GET #1,IX%(I)
4420 IF X$=LEFT$(ZN$(1),L) THEN PS=I:RANG=IX%(I):Q=1:RETURN
4430 W=I+1:GOTO 4350
4440 '==========================================================
```

ANNEXE I

DIFFERENCES ENTRE BASICS TRS-80 ET MICROSOFT 5.

	TRS-80	MICROSOFT 5.
Noms de variables	2 lettres significatives PRINT RA ⇔ PRINT RANG	40 lettres significatives
Séparateurs	Il n'y a pas d'espaces séparateurs Corollaire : Les noms de variables ne doivent pas comporter de MOT-CLE du BASIC	Les espaces séparateurs sont obligatoires
INPUT "Message";X	Si l'opérateur appuie sur <ENTER>, X conserve son ancienne valeur. Un point d'interrogation est imprimé après le message.	Une variable prend la valeur nulle si l'opérateur appuie sur RC
Adressage curseur	PRINT Y*64+X,"MESSAGE"	N'existe pas directement
Graphique	SET(X,Y) allume le point X,Y RESET(X,Y) éteint le point X,Y POINT(X,Y) teste si le point X,Y est allumé	N'existe pas en standard
Fin de fichier	LOF(n° fichier) fournit nombre d'enregistrements d'un fichier	LOF(n° fichier) fournit le nombre d'enregistrements MODULO 128
SWAP X,Y	N'existe pas : faire : A=X:X=Y:Y=A	Echange les valeurs de X et Y
ERASE table	N'existe pas : faire : FOR I=1 TO N:A(I)=0: NEXT I	Ne fonctionne qu'en interprété

A N N E X E I I

COMMANDES ESSENTIELLES DU D.O.S. TRS-80

COPY

Permet de copier des fichiers :

1/ COPY :0 TO :1 05/30/81

copie tous les fichiers de la disquette du lecteur 0 sur la disquette du lecteur 1
La date est spécifiée par MM/JJ/AA

La copie est précédée d'un formattage.
La disquette destination peut donc être vierge

2/ COPY :0 TO :0 05/30/81

permet la duplication d'une disquette avec un seul lecteur
Les disquettes 'source' et 'destination' doivent être montées alternativement.

3/ COPY XX:0 TO XX:1

copie le fichier 'XX' de la disquette du lecteur 0 sur la disquette du lecteur 1

4/ COPY XX:0 TO :1

forme abrégée

5/ COPY XX:1 TO :0

Seule la disquette destination contient le D.O.S.

6/ COPY :0 XX TO XX

Avec un seul lecteur.
Si la disquette source ne contient pas le système, placer une disquette système, frapper la commande et attendre les instructions

DIR

DIR

fournit la liste des fichiers sur la disquette du lecteur 0

DIR :1

fournit la liste des fichiers sur la disquette du lecteur 1

DIR :0(S)

fournit la liste des fichiers systèmes et des fichiers qui ne sont pas invisibles

DIR :0(I)

fournit les fichiers invisibles plus ceux n'appartenant pas au système

DIR :0(A)

fournit l'espace occupé par chaque fichier

DIR :0(S,I,A)

combine les effets de S,I,A

FREE

indique la place libre (en granules) pour les disquettes connectées
(1 granule = 1024 octets)

FORMAT

formatte une disquette

a) frapper FORMAT et suivre les instructions

b) frapper directement : FORMAT numéro lecteur,MM/JJ/AA, mot de passe

KILL nom de fichier

supprime un fichier

KILL XX:1 supprime le fichier XX sur lecteur 1

RENAME *ancien nom TO nouveau nom*

RENAME XX TO YY le fichier XX devient le fichier YY

BASIC

permet de travailler avec le système BASIC

1) BASIC connecte sous BASIC

2) BASIC * connecte sous BASIC avec le contenu de la mémoire au moment où BASIC a été quitté (par CMD "S")

3) BASIC commande BASIC

BASIC RUN "FACTU/BAS"

COMMANDES ESSENTIELLES SOUS BASIC DU D.O.S. TRS-80

(Ces commandes peuvent aussi être utilisées comme instructions BASIC)

LOAD "nom programme"

charge le programme spécifié en mémoire centrale

LOAD "FACTU/BAS" charge le programme FACTU/BAS

RUN

exécute le programme en mémoire centrale

RUN "FACTU/BAS" charge puis exécute le programme FACTU/BAS

RUN "FACTU/BAS",R charge et exécute FACTU/BAS mais ne clot pas les fichiers déjà ouverts

KILL "nom de fichier"

supprime le fichier spécifié (qui doit être clos)

MERGE "nom de programme"

concatène le programme spécifié (sauvegardé en ASCII) au programme en mémoire centrale

RENUM nouveau numéro de ligne, incrément, numéro départ, numéro fin

renumérote un programme

RENUM 100 renumérote à partir de 100 de 10 en 10

RENUM 100,5 renumérote à partir de 100 de 5 en 5

REF

1/ REF *
fournit la liste à l'écran des références de toutes les variables

2/ REF $
imprime la liste des références de toutes les variables

3/ REF variable
affiche à l'écran les références de la variable spécifiée

4/ REF * variable

affiche à l'écran les références des variables à partir de la variable spécifiée

5/ REF $ variable

imprime les références des variables à partir de la variable spécifiée

SAVE

SAVE "FACTU/BAS"	sauvegarde le programme en mémoire sous le nom de FACTU/BAS
SAVE "FACTU/TXT",A	sauvegarde le programme sous forme ASCII

CMD

Sous BASIC, permet d'accéder aux commandes D.O.S.

CMD "DIR" fournit la liste des programmes

CMD "S" retour au D.O.S.

DIRCHECK

fournit la liste des fichiers sur l'écran ou l'imprimante

ANNEXE III

QUELQUES COMMANDES CPM

DIR

```
>DIR A:
 fournit la liste des programmes sur la disquette du lecteur A
```

STAT

```
>STAT
 fournit l'espace disque de l'unité A

>STAT B:
 fournit l'espace disque de l'unité B

>STAT B:*.BAS
 fournit la liste des programmes BASIC
```

PIP

```
>PIP A:PAYE.BAS=B:PAIE.BAS
 copie le programme PAIE de l'unité B sur l'unité A sous
 le nom de PAYE

>PIP B:A:*.*
 copie tous les fichiers de A sur B

>PIP B:=*.COM
 copie toutes les commandes de l'unité A sur l'unité B

>PIP <retour chariot>

*B:=A:*.* transfère tous les fichiers de A sur B

*<retour chariot>
```

ERA

```
>ERA A:PAYE.BAS
 efface le programme PAYE
```

REN

```
>REN PAYE.BAS=PAIE.BAS
 PAIE devient PAYE
```

Achevé d'imprimer en juin 1981
sur les presses de l'imprimerie Laballery et C[ie]
58500 Clamecy
Dépôt légal : 2[e] trimestre 1981
N[o] d'édition : 86595-23-1
N[o] d'imprimeur : 20107
ISBN 2-86595-023-9



Le BASIC et ses Fichiers

Ce second tome du Basic et ses fichiers, comme le premier, est destiné aux utilisateurs de PSI disposant du Basic Microsoft : TRS-80 et systèmes fonctionnant sous CPM. Les méthodes d'accès par clé déjà abordées dans le tome I sont traitées de façon plus approfondie. Le corps de l'ouvrage est consacré à des programmes, utilitaires comme le générateur de saisie d'écran ou le tri rapide, de gestion comme la facturation ou la paie.

Editions du P.S.I.
Boîte postale 86
77400 Lagny/Marne

ISBN : 2-86595-023-9

Imprimé en France


Jacques Boisgontier
P.S.I.
Le BASIC et ses Fichiers
Tome 2